中国研究生创新实践系列大赛

“华为杯”第十七届中国研究生数学建模竞赛

学　　校	上海交通大学
参赛队号	20102480125
队员姓名	1.郭梦裕
	2.霍洋洋
	3.胡令矿

中国研究生创新实践系列大赛
“华为杯”第十七届中国研究生数学建模竞赛

题 目　质心平衡供油策略设计：贪心和搜索的结合

摘　　要：

在飞行器飞行过程中，其质心变化对飞行器的控制有着重要的影响，而各个油箱内油量的分布和供油策略将对飞行器的质心变化产生重要影响。因此，为了最小化飞行过程中的实际质心和理想质心的最大偏差，本文根据计划耗油速度和理想质心对供油策略和油量分布进行求解。具体地，本文从四个角度对飞行器质心偏差算法进行说明：对飞行器存在俯仰角的情况下的质心进行建模求解，对给定初始油量，耗油速度和理想质心对供油策略进行建模求解，对给定剩余油量对初始和剩余油量分布进行建模求解，对固定理想质心为飞行器坐标系原点，且飞行器存在俯仰角的情况下对供油策略进行建模求解。

针对问题 1，对飞行器存在俯仰角的情况下的质心进行建模求解。本文依据数学几何知识，在每个油箱的中心建立坐标系，求解每个油箱各自的质心，然后进行坐标系转换得到飞行器坐标系下的油箱质心位置，再将油箱视为质点，求解得到飞行器的质心位置。

针对问题 2，对给定初始油量，耗油速度和理想质心对供油策略进行建模求解。本文利用贪心的思路设计逐时间节点的供油策略最优算法，并忽略连续供油的限制得到最优解，再通过数据分析得到规律，利用聚类的方法修正最优解，迭代得到满足所有约束的最优可行解。为了得到更少的质心偏差，本文允许飞行器耗油存在一定程度的浪费。在允许浪费 20%计划耗油的基础上，本文通过逐时间节点的供油策略最优算法和利用聚类方法最优解修正，得到完整的供油策略，其中飞行过程中飞行器实际质心和理想质心的欧氏距离最大值为 6.95978×10^{-5} 米，其最大值在第 5365 秒产生，而飞行过程的整体耗油为 6441.746226 千克，相比于计划耗油 6441.524212 千克，仅浪费 0.222014 千克。而在相同条件和算法下，不允许浪费的供油策略得到的欧氏距离最大值为 0.1246 米，其最大值在第 4159 秒产生，相较于允许浪费时的情况，增大了约 1790 倍。因此允许耗油的策略以少量的燃油浪费带来了质心偏移的大幅度降低，更值得被采用。

针对问题 3，不仅需要对给定剩余油量对初始和剩余油量分布进行建模求解，还需要根据耗油速度和理想质心进一步地对供油策略进行建模求解。本文从剩余油量的分布进行求解，反向推导完整飞行过程的供油策略，并根据最大质心偏移的油量数据修正剩余油量分布，直到得到最终的油量分布和供油策略。根据 $t=7200s$ 时刻数据，剩余油量的初始解为(0,54, 137, 111, 120, 428)千克，通过迭代修正为(0.0, 47.18759, 131.5116, 144.8741, 130.9014, 395.5253)千克。通过反向推导得到油箱 1-6 初始化的油量为(147.50475, 1214.0329, 2019.6, 2254.2, 1560.29485, 1020)千克。此时飞行整体耗油 7298.005094kg，计划耗油 6805.174669kg，合计浪费约 7.2%。飞行器质心与理想质心距离的最大值为 0.119036m，发生在第 5372 秒。

针对问题 4，对固定理想质心为飞行器坐标系原点，且飞行器存在俯仰角的情况下对供油策略进行建模求解。由于俯仰角的存在，质心计算公式从线性关系的计算中变成了分段式非线性计算，利用问题 2 的求解思路难以实现。对于不同的俯仰角，对于 y 轴和 z 轴的计算均使用平飞状态近似，引入最大的误差为 1.8%，在误差允许的情况下能极大的简化计算，提高计算速度。另一方面，将完整的飞行过程按 60 秒为一个时间段划分，强制每一个时间段使用同样的油箱供油，其油箱使用方案有 34 种。求解采用不同方案 60 秒供油策略，选择质心偏差和最小的方案作为该 60 秒的最终供油策略，使之在可行解的基础上寻找最优解。最终得到的实际质心与理想质心的欧氏距离最大值为 0.038594 米，这一最大距离产生在第 3839 秒。其中飞行过程总消耗油量 7565.579365 千克，相比于计划耗油 7035.545163 千克，浪费 7.53%。

目　录

1.问题重述

1.1 背景介绍与问题描述

资料显示，飞行器质心对飞行安全会产生很大影响，因为设备布置问题，飞行器油箱一般分散布置在机身各处[1-2]。携带有多个油箱的飞行器在飞行过程中，通过若干个油箱联合供油以同时满足飞行任务要求和发动机工作需求。各个油箱内油量的分布和供油策略将导致飞行器质心变化，进而影响对飞行器的控制。统计资料显示，大多数此类飞行器在日常工作中都存在燃油消耗不平衡的现象[3]，因此，需要制定利于飞行器控制的各油箱的供油策略。

飞行器的结构（如油箱的位置、形状、尺寸、供油关系、供油速度限制等）影响油箱的供油策略和飞行器的质心变化。已知飞行器净质量、油箱中心相对飞行器不载油时质心的位置、各个油箱的尺寸。本问题研究的飞行器型号一共有 6 个油箱，各油箱供油示意图如图 1-1 所示。

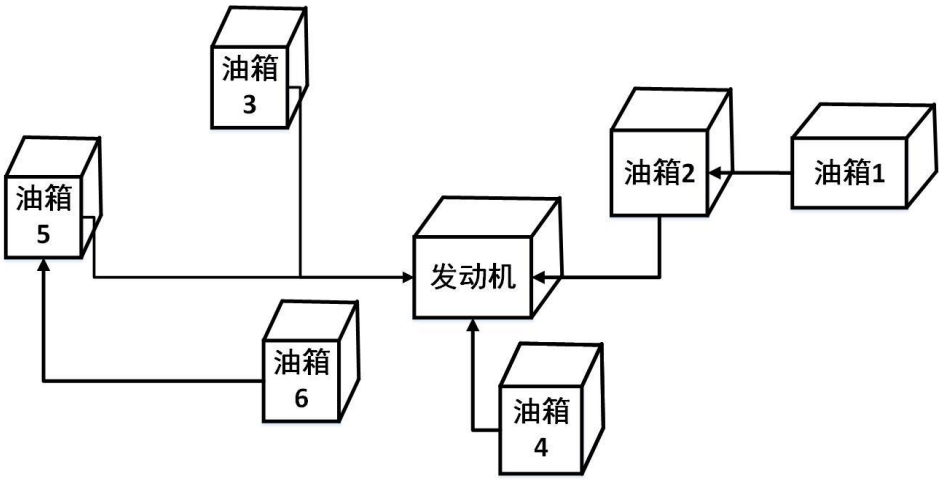

图 1-1 各油箱供油示意图

1.2 问题提出

基于已知信息建立数学模型，在给定的数据下，求解以下 4 个问题：

问题 1：已知某次任务中飞行器的 6 个油箱的供油速度及飞行器在飞行过程中俯仰角变化数据，根据 6 个油箱的初始油量，求解飞行器在任务执行过程中的质心变化曲线，并计算其质心在飞行器坐标系下的位置数据。

问题 2：已知某次飞行任务中飞行器计划耗油速度数据，与飞行器在飞行器坐标系下的理想质心位置数据。根据任务需求，在飞行器始终保持平飞的任务规划过程中，指定满足条件的 6 个油箱供油策略，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小, i.e,

$$\min \max_t \| \vec{c}_1(t) - \vec{c}_2(t) \|_2$$

计算出飞行器飞行过程中 6 个油箱各自的供油速度和 4 个主油箱的总供油速度、以及飞行器瞬时质心与理想质心距离的最大值和 4 个主油箱的总供油量。

问题 3：假定初始油量未定，已知某次任务的飞行器计划耗油速度数据，与飞行器在飞行器坐标系下的理想质心位置数据。在飞行器始终保持平飞(俯仰角为 0)的任务规划过程中，制定飞行器在该次任务满足条件的 6 个油箱初始载油量及供油策略，使得本次任务结束时 6 个油箱剩余燃油总量至少 $1\,m^3$，并且飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小, i.e,

$$\min \max_t \| \vec{c}_1(t) - \vec{c}_2(t) \|_2$$

给出 6 个油箱的初始载油量、飞行器飞行过程中 6 个油箱的供油速度和 4 个主油箱的总供油速度(时间间隔为 $1\,s$)、以及飞行器质心与理想质心距离的最大值和 4 个主油箱的总供油量。

问题 4：在实际任务规划过程中，飞行器俯仰角随时间变化。已知飞行器俯仰角的变化数据和耗油速度数据，为本次任务制定油箱供油策略，使得飞行器瞬时质心 $\vec{c}_1(t)$ 与飞行器(不载油)质心 $\vec{c}_0$ 的最大距离达到最小，即

$$\min \max_t \| \vec{c}_1(t) - \vec{c}_2(t) \|_2$$

绘出飞行器飞行过程中 6 个油箱各自的供油速度曲线，再将 4 个主油箱的总供油速度曲线(时间间隔为 $1\,s$)与计划耗油速度曲线绘于一个图中，给出飞行器瞬时质心与飞行器(不载油)质心 $\vec{c}_0$ 的最大距离偏差以及 4 个主油箱的总供油量。

2.模型假设

- 假设 1：可以将油箱简化为长方体且固定在飞行器内部，油箱的长、宽、高的三个方向与飞行器坐标系的 x, y, z 轴三个方向平行。
- 假设 2：以飞行器不载油时的质心 $\vec{c}_0$，为原点建立惯性坐标系、飞行器坐标系；
- 假设 3：第 i 个油箱的供油速度上限为 U_i $(U_i > 0)$，每个油箱一次供油的持续时间不少于 60 秒。
- 假设 4：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油。
- 假设 5：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油。
- 假设 6：飞行器在执行任务过程中，各油箱联合供油的总量应至少满足发动机的对耗油量的需要，若某时刻供油量大于计划耗油量，多余的燃油可通过其它装置排出飞行器。
- 假设 7：飞行器的飞行过程中只有俯仰和平飞的情况。飞行器的俯仰将导致各油箱相对地面的姿态发生偏斜，在重力作用下，油箱的燃油分布随之变化。
- 假设 8：飞行器飞行过程是平稳的，加速度为 0，即油箱内燃油液面始终与水平面平行。
- 假设 9：油箱中不会存在油珠飞溅、油珠挂壁等情况，且油箱内的燃油可以被完全使用。
- 假设 10：燃油是完全密度均匀的。
- 假设 11：假设飞行器飞行过程中，每个油箱为发动机或其他油箱供油是瞬时完成的，且输送频率为每秒 1 次（或者不输送）。

3.符号说明

符号	意义
T	油箱的集合
t_i	油箱编号
O	惯性坐标系下的原点
x，y，z	惯性坐标系下的坐标轴
$O(t)$	飞行器坐标系下的原点
$x(t)$，$y(t)$，$z(t)$	飞行器坐标系下的坐标轴
a_i	油箱i的长（飞行器坐标系x轴方向的长度）
b_i	油箱i的宽（飞行器坐标系y轴方向的长度）
c_i	油箱i的高（飞行器坐标系z轴方向的长度）
$\vec{c}_0$	飞行器不载油时的质心位置
$\vec{c}_1(t)$	飞机时刻t的实际质心位置
$\vec{c}_2(t)$	飞机时刻t的理想质心位置
M	飞行器净重量
$\vec{P}_i$	第i个空油箱的中心位置
U_i	第i个油箱的供油速度上限
$v_i(t)$	第i个油箱时刻t的供油速度
$V(t)$	飞行器时刻t的耗油速度
$\vec{p}_i(t)$	第i个油箱内的燃油在时刻t的质心位置
$S_i(t)$	第i个油箱内的燃油在时刻t在xOz
$m_{i\max}$	第i个油箱能够容纳的燃油质量上限
$m_i(t)$	第i个油箱时刻t的剩余油量

4. 问题 1 模型的建立与求解

4.1 问题 1 描述与分析

问题 1 给出了某次任务中飞行器的 6 个油箱的供油速度及飞行器在飞行过程中的俯仰角变化数据，根据飞行器的简化模型和假设条件，建立物理模型对不同情况作分类讨论即可解决问题，如图 4-1 给出了根据题中所给数据得到的油箱相对位置图。

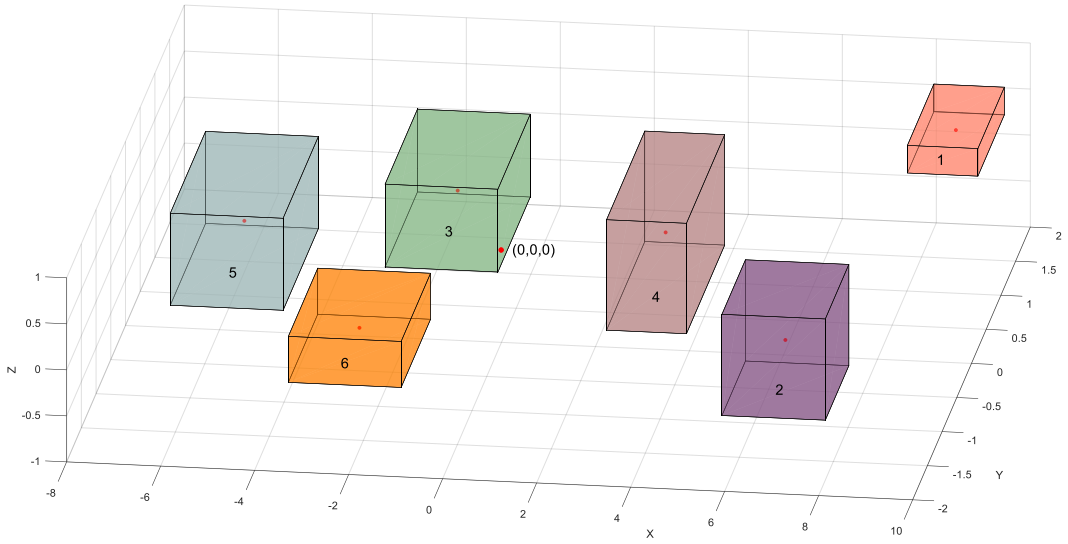

图 4- 1 油箱三维位置示意图

为了分别求解每个油箱在时间 t 的质心位置，对每个油箱分别建立坐标系。以一个油箱为例，以其在中心位置为原点，建立坐标系 $O-x'y'z'$，如图 4-2 所示。

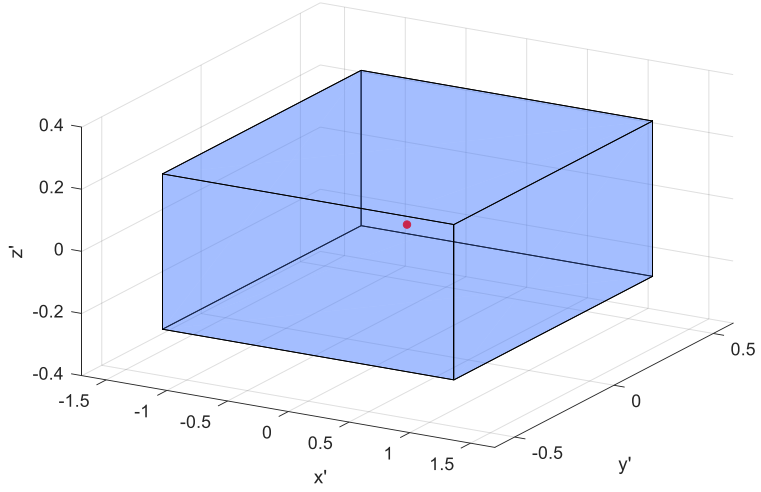

图 4- 2 三维油箱坐标系示意图

由于不考虑飞行器的左右偏转，故油箱内的燃油形状始终是柱体，在飞行器俯仰角不同、油箱内燃油体积不同的情况下，其形态可以分为三棱柱、四棱柱、五棱柱三种情况。由于燃油是密度均匀的，故在油箱坐标系内，无论燃油处于何种形态，y'方向上的坐标始终为0，可以不用考虑。另一方面，燃油棱柱质心在x'方向和z'方向上的坐标与棱柱在$x'Oz'$平面上的截面的质心在x'方向和z'方向上的坐标相同，因此，我们可以将问题进一步简化为求解燃油截面图形的质心。

因此将单一油箱的三维质心问题降维成为xOz平面上求解多边形二维质心问题。根据油箱中所剩余油量不同，共存在三种不同的二维图形，分别为三角形，四边形，五边形。以飞机向上仰冲时为例，示意图如图4-3所示。

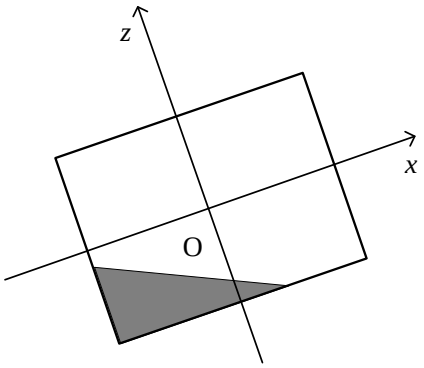

图4-3 所剩油量为三角形示意图

当油箱中所剩余油量较少时，会在油箱的xOz平面形成如上图4-3所示的三角形，假设油是质量分布均匀且稳定的情况下。此时油箱内油的质心是三角形油面的质心。

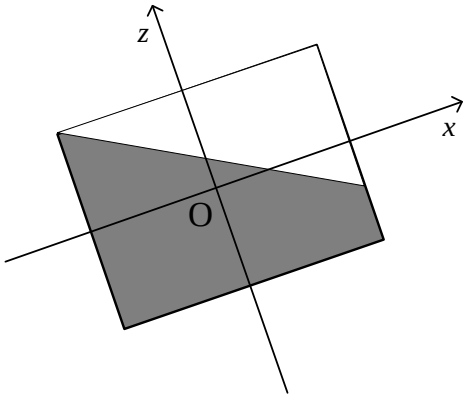

图4-4 所剩油量为四边形示意图

当油箱中所剩余油量较少时，会在油箱的xOz平面形成如图4-4所示的四边形，假设油是质量分布均匀且稳定的情况下。此时油箱内油的质心是四边形油面的质心。

同时若油箱内油量继续增加，则会在油箱内形成五边形的截面，其示意图如下图4-5所示。

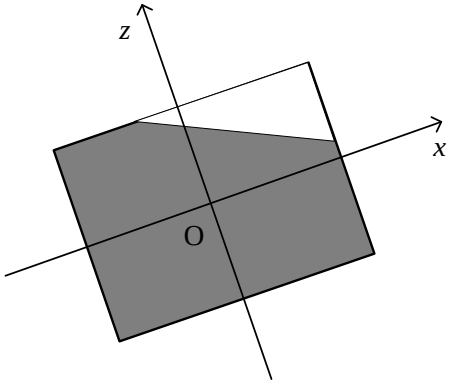

图 4- 5 所剩油量为五边形示意图

根据质心定义，针对四边形及五边形的质心求解此次建模采用将多边形三角化算法。多边形三角化是对多边形的基本处理，它是指在原有图形的基础上将多边形划分成许多互不重叠的三角形[4]。三角化一般包括平面点集三角化和多边形三角化，最经典的 Delaunay 三角剖分算法采用的是平面点集三角化方法，然后根据三角形的质心求解进行加权平均，最终解得多边形的质心。

故针对问题一的整体思路是将六个油箱单独建立坐标系，以油箱不装油量时在 x'O'z' 平面上的质心为坐标原点，建立与飞机坐标系平行的油箱坐标系。进而分别计算不用时刻的油箱质心变化，用质心代替油箱，将问题化简为点系统下的质心。最后解得整个系统的质心。

4.2 多边形三角化求解质心

在求解二维油箱截面模型质心时需要计算三角形、四边形、五边形的质心。三角形质心为其顶点坐标之和的平均值。但是四边形和五边形不能直接用顶点坐标之和的平均值求得。则其中一个求解多边形质心的方案是多边形三角化。

因为问题一中所涉及的只是规则的四边形和五边形，则三角化的方法也可以进行简化，以四边形为例，其分解示意图如 4-6 所示。

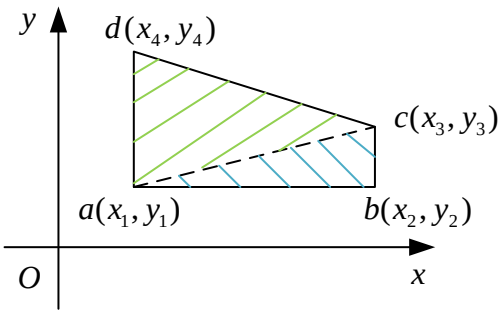

图 4- 6 四边形三角化示意图

如上图 4-4 所示，四边形 abcd 被黑色虚线部分分解成为三角形 abc 和三角形 acd。通过三角形质心公式，分别求得三角形 abc 的质心为：

$$(\frac{x_1+x_2+x_3}{3},\frac{y_1+y_2+y_3}{3}) \tag{4-1}$$

三角形 acd 的质心为：

$$(\frac{x_1+x_3+x_4}{3},\frac{y_1+y_3+y_4}{3}) \tag{4-2}$$

然后分别求得三角形 abc 的面积 S_1 及三角形 acd 的面积 S_2。

最终求得四边形 abcd 的质心为：

$$(\frac{(x_1+x_2+x_3)*S_1+(x_1+x_3+x_4)*S_2}{3*(S_1+S_2)},\frac{(y_1+y_2+y_3)*S_1+(y_1+y_3+y_4)*S_2}{3*(S_1+S_2)}) \tag{4-3}$$

五边形的质心求解同上，将五边形分成三个三角形进行质心的加权平均。

4.3 二维油箱子模型质心求解建模

将三维的油箱模型降维成为 *xOz* 平面上的二维平面模型。当油箱中加入初始化的油量之后，其示意图如下所示。

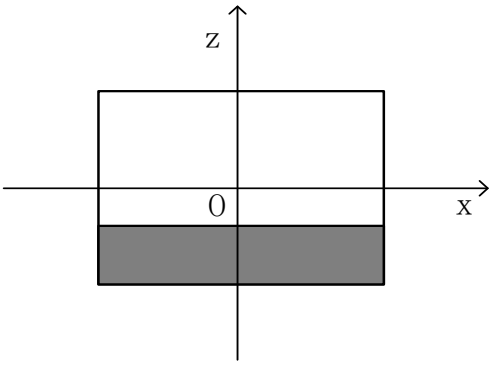

图 4- 7 二维截面油箱示意图

根据油箱 *xOz* 平面的油箱长和高的比例不同，会导致在某一相同倾斜角度下，在油箱内所剩余油相同时，进入不同的油分布情况。根据计算，阈值的倾斜角度为 $\arctan\left(\frac{l}{h}\right)$，其中 *h* 为油箱高度，即沿 *z* 轴方向的高度，*l* 为油箱长度，即沿 *x* 轴方向的长度。

同时根据飞行器的上下俯冲、直飞的不同情况。所以将二维油箱模型质心求解建模划分为三类，分别为上仰、俯冲和直飞，六种油面形状，共有 2*6+1=13 种模型。同时因为当飞行器姿态为上仰和俯冲时，油面形状关于 *z* 轴对称。

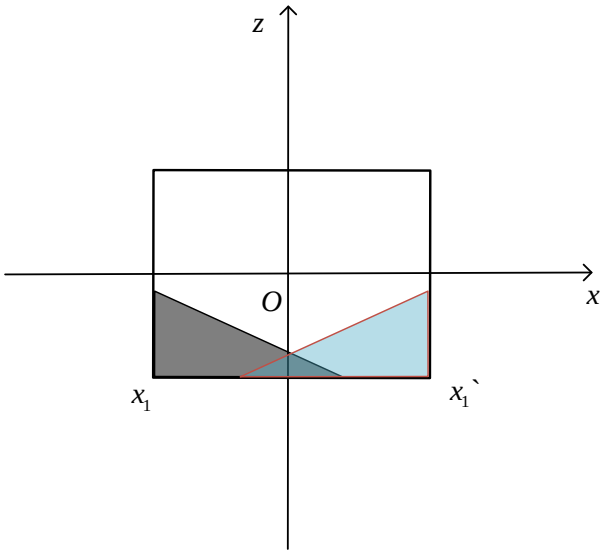

图 4- 8 上仰和俯冲坐标示意图

假设上图所示油箱模型长度l，高度为h，如上图中点x_1在黑色所示坐标系下的坐标为$(-\frac{l}{2},-\frac{h}{2})$,则$x_1'$在图中红色坐标系下的坐标为$(\frac{l}{2},-\frac{h}{2})$，因此飞行器为上仰时的任意顶点坐标与俯冲时的对应顶点坐标关于z轴对称，因此在建模求解平面质心时，只需求解上仰或者俯冲中的任意情况，另一种情况下的对应质心为该情况下的质心坐标的z轴对称坐标值。

基于如上推论，以下建模只分析仰角时的模型。

根据上文中计算得到阈值的倾斜角度为$\theta_t=\arctan\left(\frac{l}{h}\right)$。

当倾斜角度$\theta\le\arctan(\frac{l}{h})$时，根据油箱中所剩余的油量不同，共存在三种平面模型：三角形、四边形、五边形。存在两种边界情况，其边界示意图 4-9 所示。

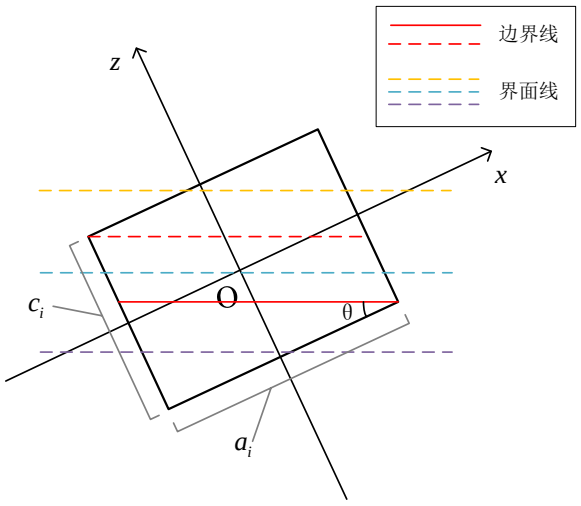

图 4- 9 上仰边界示意图

上图 4- 9 中的红色线段为图形边界，包括实线与虚线，图中黄色、蓝色、紫色虚线段分别示意不同的界面情况，分别对应油箱内截面图形为五边形、四边形、三角形时。

首先根据油箱尺寸求解出两条红色线段所代表的油箱内所剩余油量的截面面积的边界值。根据图中所示倾斜角度 θ 及图中所示油箱第 i 个油箱的尺寸为 a_i 及 b_i。可以求得三角形与四边形的面积边界值 S_1 为：

$$S_1 = \frac{1}{2} * a_i^{\,2} * \tan\theta \tag{4-4}$$

四边形与五边形的面积边界值 S_2 为：

$$S_2 = a_i * c_i - \frac{1}{2} * a_i^{\,2} * \tan\theta \tag{4-5}$$

1) 当 $S \le S_1$ 时，此时截面模型如图 4-10 所示。

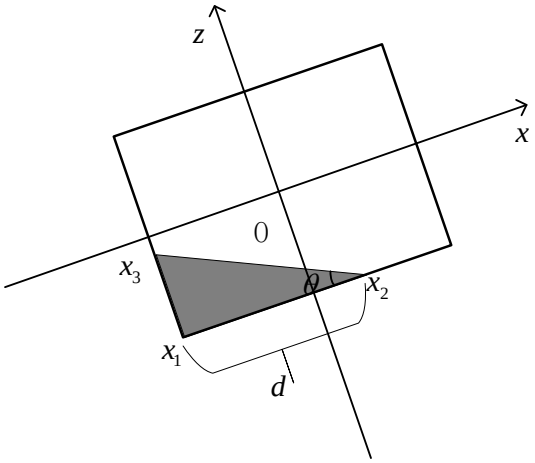

图 4- 10 三角形截面坐标示意图

其中 $d = \sqrt{\frac{2 * S_i}{\tan\theta}}$，$S_i$ 为第 i 个油箱中所剩余油量截面面积 $S_i = \frac{V_i}{b_i}$，其中 $V_i = \frac{m_{i(t)}}{\rho}$，$\rho$ 为油的密度 $850\,kg/m^3$。根据所求 d 求解得出点 x_1、x_2、x_3 坐标。

$$\begin{aligned} x_1 &= (-\frac{a_i}{2}, -\frac{c_i}{2}) \\ x_2 &= (-\frac{a_i}{2}, d * \tan\theta - \frac{c_i}{2}) \\ x_3 &= (d - \frac{a_i}{2}, -\frac{c_i}{2}) \end{aligned} \tag{4-6}$$

因此得出此时截面中三角形的质心坐标为 $\frac{x_1 + x_2 + x_3}{3}$。即三个点的平均坐标值。

2) 当 $S_1 < S \le S_2$ 时，此时截面模型如图 4-11 所示。

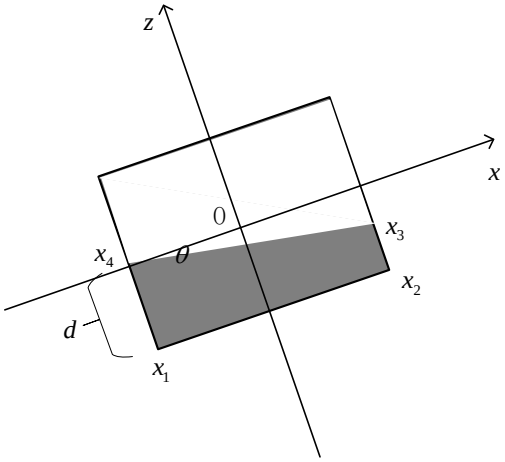

图 4- 11 四边形截面坐标示意图

其中 $d=\frac{S}{a_i}+\frac{1}{2}*a_i*\tan\theta$，$S_i$ 为第 i 个油箱中所剩余油量截面面积 $S_i=\frac{V_i}{b_i}$，其中 $V_i=\frac{m_{i(t)}}{\rho}$，$\rho$ 为油的密度 $850\,kg/m^3$。根据所求 d 求解得出点 x_1、x_2、x_3、x_4 坐标。

$$\begin{aligned} x_1&=(-\frac{a_i}{2},-\frac{c_i}{2})\\ x_2&=(\frac{a_i}{2},-\frac{c_i}{2})\\ x_3&=(\frac{a_i}{2},-\frac{c_i}{2}+d-a_i*\tan\theta)\\ x_4&=(-\frac{a_i}{2},d-\frac{c_i}{2}) \end{aligned} \tag{4-7}$$

因此根据上文推导四边形三角化算法得出该模型下的截面质心位置。

3) 当 $S>S_2$ 时，此时截面模型为：

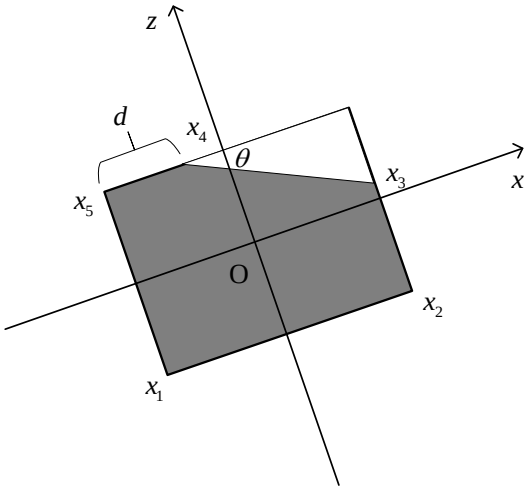

图 4- 12 五边形截面坐标示意图

其中 $d = a_i - \sqrt{\frac{2(a_i * c_i - S_i)}{\tan\theta}}$，$S_i$ 为第 i 个油箱中所剩余油量截面面积 $S_i = \frac{V_i}{b_i}$，其中 $V_i = \frac{m_{i(t)}}{\rho}$，$\rho$ 为油的密度 $850\,kg/m^3$。根据所求 d 求解得出点 x_1、x_2、x_3、x_4、x_5 坐标。

$$
\begin{aligned}
x_1 &= (-\frac{a_i}{2}, -\frac{c_i}{2}) \\
x_2 &= (\frac{a_i}{2}, -\frac{c_i}{2}) \\
x_3 &= (\frac{a_i}{2}, \frac{c_i}{2} - (a_i - d) * \tan\theta) \\
x_4 &= (-\frac{a_i}{2} + d, \frac{c_i}{2}) \\
x_5 &= (-\frac{a_i}{2}, \frac{c_i}{2})
\end{aligned}
\tag{4-8}
$$

因此根据上文推导五边形三角化算法得出该模型下的截面质心位置。

当倾斜角度 $\theta > \arctan(\frac{l}{h})$ 时，根据油箱中所剩余的油量不同，同样共存在三种平面模型：三角形、四边形、五边形。存在两种边界情况，其边界示意图与 $\theta \le \arctan(\frac{l}{h})$ 时存在差别，如下图 4-13 所示。

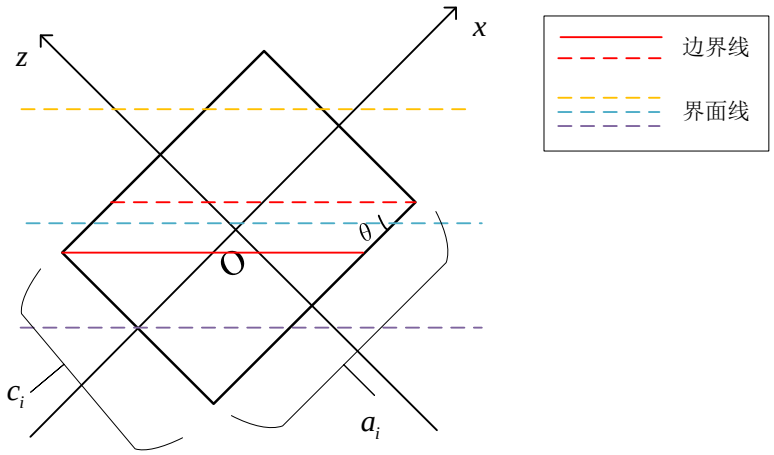

图 4- 13 上仰边界示意图

上图 4-13 中的红色线段为图形边界，包括实线与虚线，图中黄色、蓝色、紫色虚线段分别示意不同的界面情况，分别对应油箱内截面图形为五边形、四边形、三角形时。

同上文分析相同，首先根据油箱尺寸求解出两条红色线段所代表的油箱内所剩余油量的截面面积的边界值。根据图中所示倾斜角度 θ 及图中所示油箱第 i 个油箱的尺寸为 a_i 及 b_i。可以求得三角形与四边形的面积边界值 S_1 为：

$$S_1 = \frac{1}{2} * \frac{c_i^{\,2}}{\tan\theta} \tag{4-9}$$

四边形与五边形的面积边界值S_2为：

$$S_2 = a_i * c_i - \frac{1}{2} * \frac{c_i^{\,2}}{\tan\theta} \tag{4-10}$$

4) 当$S \le S_1$时，同上文 1)情况分析计算结果相同。

5) 当$S_1 < S \le S_2$时，此时截面模型不同于上文[2]所计算模型，因此需要重新建立模型，其示意图如下所示。

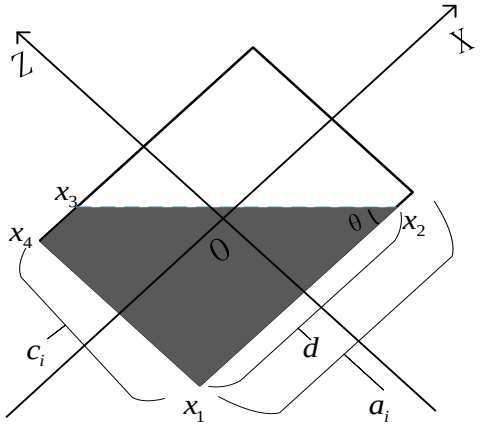

图 4- 14 四边形截面坐标示意图

其中$d = \frac{S}{c_i} + \frac{c_i}{2 * \tan\theta}$，$S_i$为第 i 个油箱中所剩余油量截面面积$S_i = \frac{V_i}{b_i}$，其中$V_i = \frac{m_{i(t)}}{\rho}$，$\rho$为油的密度$850\,kg/m^3$。根据所求$d$求解得出点$x_1$、$x_2$、$x_3$、$x_4$坐标。

$$\begin{aligned} x_1 &= (-\frac{a_i}{2}, -\frac{c_i}{2}) \\ x_2 &= (d - \frac{a_i}{2}, -\frac{c_i}{2}) \\ x_3 &= (-\frac{a_i}{2} + x - \frac{c}{\tan\theta}, \frac{c_i}{2}) \\ x_4 &= (-\frac{a_i}{2}, \frac{c_i}{2}) \end{aligned} \tag{4-11}$$

因此根据上文推导四边形三角化算法得出该模型下的截面质心位置。

6) 当$S > S_2$时，同上文[3]情况分析计算结果相同。

当飞机进行平飞时，$\theta = 0$，油箱内的油在 xOz 平面呈现长方形，其示意图如下所示。

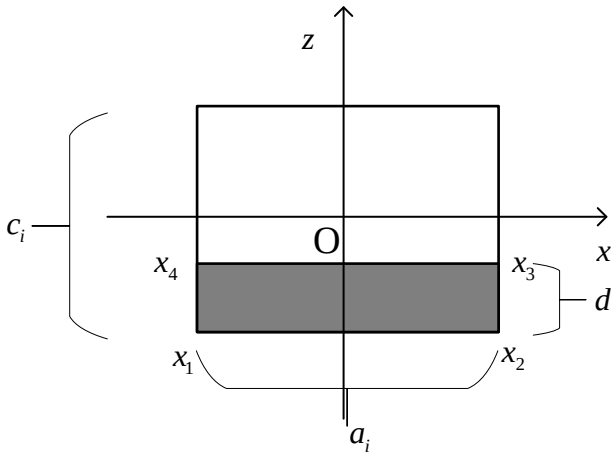

图 4- 15 平飞油面示意图

其中 $d=\frac{S}{a_i}$， S_i 为第 i 个油箱中所剩余油量截面面积 $S_i=\frac{V_i}{b_i}$，其中 $V_i=\frac{m_{i(t)}}{\rho}$， ρ 为油的密度 $850\,kg/m^3$。根据所求 d 求解得出点 x_1、 x_2、 x_3、 x_4 坐标。

$$\begin{aligned} x_1&=(-\frac{a_i}{2},-\frac{c_i}{2})\\ x_2&=(\frac{a_i}{2},-\frac{c_i}{2})\\ x_3&=(\frac{a_i}{2},d-\frac{c_i}{2})\\ x_4&=(-\frac{a_i}{2},d-\frac{c_i}{2}) \end{aligned} \tag{4-12}$$

则长方形的质心为 $(0,\frac{d}{2}-\frac{c_i}{2})$。

4.4 三维油箱质心坐标求解

在 4.2 节中分析求解了二维 xOz 平面的油箱模型质心，同时 y 轴方向质心在自身油箱坐标系中的值为 0。至此完成了每个油箱相对于自身坐标系的建模。

4.5 三维飞行器质心模型

油箱相对自身的坐标系以自身中心为原点，所以其相对于飞行器坐标系的偏移量为附件 1 中的油箱中心位置。因此每个三维油箱质心在飞行器坐标系中的坐标为三维坐标加偏移量。假设第 i 个油箱在自身坐标系中的质心位置为 (x_i,y_i,z_i)，其油箱的中心位置为 (x_{ti},y_{ti},z_{ti})，则其在飞行器系统中的位置为 $(x_i+x_{ti},y_i+y_{ti},z_i+z_{ti})$。

将飞行器系统分解为不加油时的整体飞机系统（包括油箱），其质心为(0,0,0)，及六个油箱中的油模型，其质心为上节 4.3、4.4 中求解可得，第 i 个油的质心设为 (x_i,y_i,z_i),则飞机器（含油）整体的质心 (x,y,z) 求解公式为：

$$(x,y,z)=\frac{\sum_{i=1}^{6}(x_i+y_i+z_i)m_{i(t)}}{M+\sum_{i=1}^{6}m_{i(t)}} \tag{4-13}$$

4.6 问题 1 结果与分析

此次分别在 matlab 与 Visual Studio 2019 通过 C++求解整体质心偏移模型。两种方式解得数据相同。解得质心在 x 轴、y 轴、z 轴的偏移情况如图 4-16 所示。

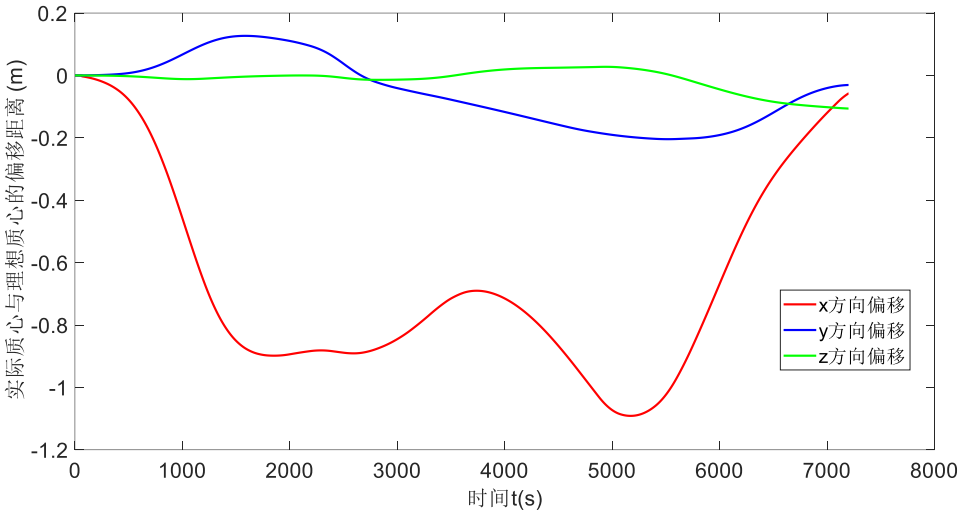

图 4- 16 质心各方向偏移示意图

根据上图所示，发现质心在 x 轴向上的偏移量最大， x 轴向的偏移对整体的偏移影响最大。y 轴方向的质心偏移仅与各个油箱所剩油量相关，在单独计算各个油箱质心时，y 轴方向质心不变，所以 y 轴质心偏移量也较小。同时 z 轴向的质心偏移最为稳定，变化幅度变化量最小。将三个方向的偏移合成一个方向，求解整体的质心偏移量示意图如图 4-17 所示。

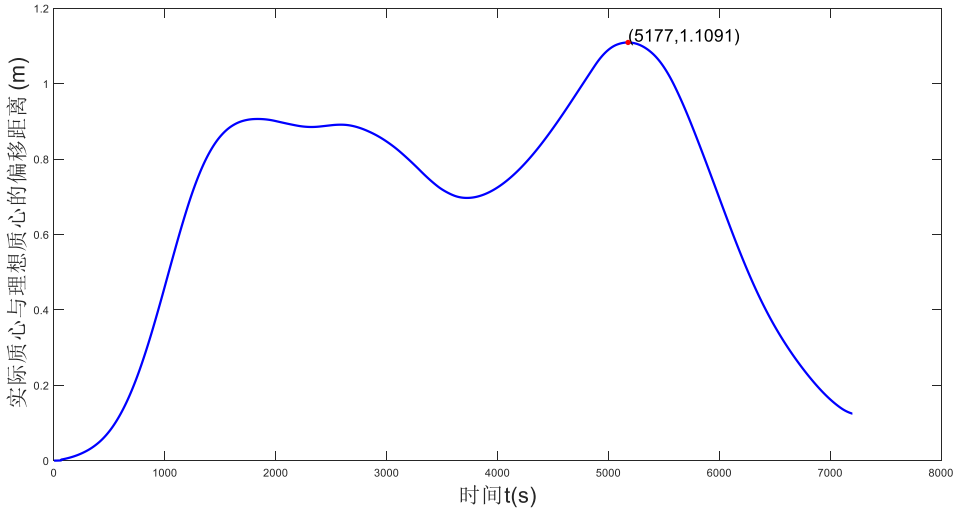

图 4- 17 质心整体偏移示意图

如上图所示，质心整体偏移呈现先上升，再下降，然后再上升，最后下降的趋势。基本与 x 轴向的偏移趋势相同。同时质心整体偏移的最大值发生在第 5177 秒，最大偏移量为 1.1091。也与 x 轴的最大值出现时刻大致吻合。同样说明 x 轴向的偏移对整体的偏移影响最大。

5、问题 2 模型的建立与求解

5.1 问题描述与分析

问题 2 要求规划满足任务需求的油箱供油策略，目标是使得飞行器每一时刻的质心位置与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。题目给定了飞行器计划耗油速度及飞行器在飞行器坐标系下的理想质心位置数据。与问题 1 不同的是，问题 2 限定了飞行器俯仰角始终为 0，简化了质心求解过程，可以认为在油箱坐标系下，油箱的质心位置只会存在 z 方向上的变化。由于飞行器每个油箱内的初始油量已经给定，我们需要确保油量始终足够飞行器完成本次飞行任务，在此基础上寻找问题的最优解。

问题 2 的目标在于使整体情况的质心偏移最小，但探索完整飞行过程中的质心偏移最大值最小需要求解大量的参数，例如每个时刻的油箱供油速度和油箱的选择，这使得求解十分困难。另一方面，在实际飞行器飞行过程中，后续飞行过程的油耗和质心偏移难以估测，因此通过当前油量和当前的油耗分析进行供油策略的设计是一个比较合理的可行方案。基于此，我们对模型进行简化，每个时刻仅根据当前时刻剩余油量和需求油量求解供油速度和油箱选择，即用每个时刻的局部最优解代替全局最优解，使得问题得以解决。

5.2 模型建立

（1）目标函数

在问题 2 中，变量是每个油箱每一时刻的供油速度，目标是使得当前时刻的燃油被消耗后，质心位置与当前时刻质心理想位置的欧氏距离最小，依此建模，目标函数如下：

$$\begin{aligned}\min f &= \|\vec{c}_1(t)-\vec{c}_2(t)\|_2 \\ &= \sqrt{(x_{real}(t)-x_{ideal}(t))^2+(y_{real}(t)-y_{ideal}(t))^2+(z_{real}(t)-z_{ideal}(t))^2} \quad \forall t\end{aligned} \tag{5-1}$$

其中$(x_{ideal}(t), y_{ideal}(t), z_{ideal}(t))$表示时刻$t$飞行器在飞行器坐标系下的理想质心坐标，$(x_{real}(t), y_{real}(t), z_{real}(t))$表示时刻$t$飞行器在飞行器坐标系下的实际质心坐标，可以用当前时刻各个油箱的质量和质心坐标求出：

$$\begin{cases} x_{real}(t)=\dfrac{\sum_{i=1}^{6} m_i(t)x_i(t)}{M+\sum_{i=1}^{6} m_i(t)x_i(t)} \\ y_{real}(t)=\dfrac{\sum_{i=1}^{6} m_i(t)y_i(t)}{M+\sum_{i=1}^{6} m_i(t)y_i(t)}, \quad \forall t \\ z_{real}(t)=\dfrac{\sum_{i=1}^{6} m_i(t)z_i(t)}{M+\sum_{i=1}^{6} m_i(t)z_i(t)} \end{cases} \tag{5-2}$$

其中$(x_i(t), y_i(t), z_i(t))$表示油箱i在t时刻时在飞行器坐标系下质心的坐标，$m_i(t)$表示油箱i在时刻t的剩余燃油质量，可以用递推式得到：

$$\begin{cases} m_i(t) = m_i(t-1) - v_i(t), i = 1,3,4,5 \\ m_2(t) = m_2(t-1) - v_2(t) + v_1(t) \\ m_5(t) = m_5(t-1) - v_5(t) + v_6(t) \end{cases} \quad \forall t \geq 1 \tag{5-3}$$

$m_i(0)$ 题目中已给定。

根据问题 1 的推导，当 $m_i(t)$ 确定后，引入俯仰角为 0 的条件，我们即可以计算出 $(x_i(t), y_i(t), z_i(t))$。

至此，我们建立起了每一时刻飞行器实际质心位置、理想质心位置的欧氏距离与油箱供油速度的函数关系，并指出了模型的目标函数。

（2）约束条件

约束 1：供油速度约束：每一时刻每个油箱的供油速度应为非负数，且不能超过油箱的供油速度上限和油箱当前的剩余油量，可以用不等式描述为：

$$0 \leq v_i(t) \leq \min(U_i, m_i(t-1)/\Delta t), \quad \forall t, \quad i = 1,2,3,4,5,6 \tag{5-4}$$

在本问题中，$\Delta t = 1s$。同时，这一约束也使得油箱的油量必定非负。

约束 2：耗油速度约束：每一时刻油箱 2、3、4、5 的供油速度之和应不小于当前时间的计划耗油速度；另一方面，我们允许 2、3、4、5 供油速度之和大于当前时间的计划耗油速度，即我们允许一定程度的油量浪费换取更小的实际质心和理想质心的偏差。进一步地，由于我们在计算时刻 t 的油箱供油速度时没有考虑之后完成之后飞行任务需要的耗油量，为了保证后续油量始终充足，使用参数 α 来描述浪费的上限：

$$V(t) \leq \sum_{i=2}^{5} v_i(t) \leq \alpha V(t), \quad \alpha \geq 1, \quad \forall t \tag{5-5}$$

约束 3：油箱容量约束：由于 6 个油箱的容量限制，每个油箱中燃油体积不能超过其能容纳的燃油质量上限 $m_{i\max}$：

$$m_{i\max} = a_i b_i c_i \times 850, \quad i = 1,2,3,4,5,6 \tag{5-6}$$

在本问题中，由于初始油量已经给定，油箱 1、3、4、6 只能消耗燃油，因此必定满足油量小于油箱容量。因此只需要考虑每一时刻油箱 1 给油箱 2 供油、油箱 6 给油箱 5 供油后，油箱 2、油箱 6 中的燃油体积不能超过其能容纳的燃油质量上限：

$$\begin{cases} m_2(t-1) + v_1(t) \leq m_{2\max} \\ m_5(t-1) + v_6(t) \leq m_{5\max} \end{cases}, \quad \forall t \tag{5-7}$$

约束 4：供油油箱数约束：供油由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油。在每个时刻 t，$p_i(t)$ 表示油箱 i 是否处于可以供油的状态，为 1 表示可以供油，为 0 表示不能供油，则有：

$$p_i(t) = 0 \text{ or } 1, \quad \text{i=1,2,3,4,5,6}$$

$$\begin{cases} \sum_{i=2}^{5} p_i(t) \leq 2 \\ \sum_{i=1}^{6} p_i(t) \leq 3 \end{cases}, \quad \forall t \tag{5-8}$$

约束 5：$p_i(t)$ 与 $v_i(t)$ 之间的约束：当 $p_i(t)=0$ 时，表示油箱 i 不能被使用，$v_i(t)$ 必定为 0。

约束 6：供油持续时间约束：每个油箱的持续供油时间不得少于 60 秒，设油箱 i 某次向发动机或其他油箱供油在 t_{i1} 时刻开始，在 t_{i2} 时刻结束，则有：

$$t_{i2}-t_{i1}\geq 60,\ \ i=1,2,3,4,5,6 \tag{5-9}$$

综上，我们完成了问题二模型的建立。

5.3 模型实现与求解

5.3.1 算法设计

如果每一时刻都考虑各油箱供油策略要有利于飞行器完成一次飞行任务的全局耗油需求，情况过于复杂。因此，我们考虑采用贪心算法的思想，逐个时间节点考虑当前情况的最优解。另一方面，约束 5 的引入使得需要对 60 秒的过程整体求解，极大的增加了求解器的压力，为了简化模型方便计算，同样引入贪心的思想，暂时搁置约束 5 的限制，先求出逐个时间节点的最优解，再对该解进行修正迭代以得到满足约束 5 的最终结果。

对于逐个时间节点的求解，我们使用了 LINGO 数学软件进行求解。LINGO 是 Linear Interactive and General Optimizer 的缩写，即“交互式的线性和通用优化求解器”，由美国 LINDO 系统公司（Lindo System Inc.）推出的，为非线性规划和线性规划提供了很好的求解方案。问题二的求解模型已有前文给出，其模型为 MINLP，即混合整数非线性规划，可以利用 LINGO 软件很好的求解。此外，题目给出的精度十分高，某些初始状态数据达到 1e-9 的量级，因此对 LINGO 的非线性求解器参数进行修改，其中 Final Nonl Feasibility Tol 设置为 1e-16，Nonlinear Optimality Tol 设置为 1e-17，并且开启相关策略加速求解和提高求解精度，其设置可参考下图进行。

在得到逐个时间节点的最优解方案后，需要对其进行修正以满足约束 5 的限制。其基于以下两个现象：

1) **数据出现“集群”现象**：如下表所示，第 95-100 秒的供油决策均倾向于 2 号油箱供油，可以认为最终方案中该时间段均由 2 号油箱供油。同时存在少数数据不完全满足该现象，例如表中的 95 时刻的数据，猜想其可能是求解器的精度和结果的不稳定性导致，将其用 2 号油箱供油的方案相比于原方案带来的误差可以容忍（1e-4 级别），并且几乎不会对整个飞行过程的实际质心与理想质心最大偏移产生影响。因此我们可以通过类似聚类的方法，对逐个时间节点的最优解方案进行修正。

表格 5- 1 部分时间节点的供油策略

时刻	1 号油箱供油速度	2 号油箱供油速度	3 号油箱供油速度	4 号油箱供油速度	5 号油箱供油速度	6 号油箱供油速度
95	0	0.009824	0.000232	0	0	0
96	0	0.01017	0	0	0	0
97	0	0.010286	0	0	0	0
98	0	0.010403	0	0	0	0
99	0	0.010521	0	0	0	0
100	0	0.01064	0	0	0	0

2) **修正带来的误差积累**：虽然现象 1 表示其单个时间节点上的修正不会带来不可容忍的偏差，但随着时间的积累，其误差存在积累而使得后续数据不可信，另外修正使得油

箱油量数据偏差大，进一步加剧了误差。因此在完成对一定时间段供油策略的修正后，需要对后续数据进行重新求解，保证后续逐时间节点最优解的可靠性。

基于以上两个现象，我们对供油方案进行修正，其流程如下所示。其中初始化供油策略即对所有油箱不设置约束，然后对求解逐个时间节点的最优解方案。再利用聚类的思路，对数据进行修正，此处我们设置了 1000 秒的时间段，即修正 0-1000 秒的供油策略，然后将前 1000 秒的供油策略固定，重新求解逐时间节点的最优解，再修正下一个 1000 秒的时间段，直到全部数据修正完成，输出供油策略。

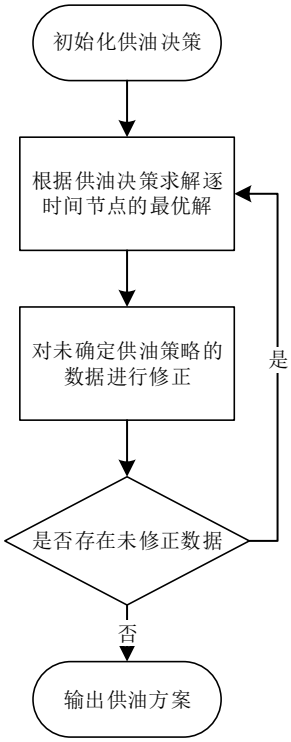

图 5- 1 问题 2 求解算法流程图

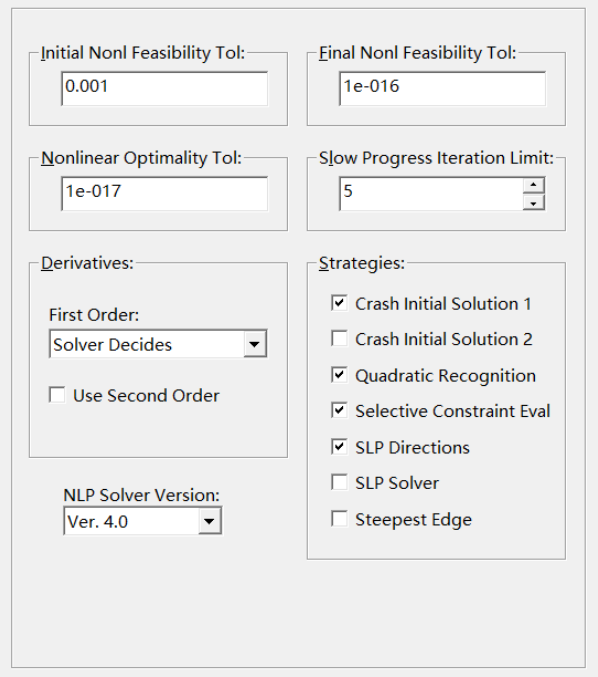

图 5- 2 问题 2 Lingo 软件求解器参数设置

5.3.2 问题 2 求解结果及分析

根据 5.3.1 节所述的算法流程，我们得到了原始的逐个时间节点的最优解方案，如图 5-3 所示，从上到下分别展示了 1-6 油箱的供油曲线。可以看到在油箱的选择上具有不稳定性，在几组近似的解中来回震荡，显示出图中的供油策略的巨大抖动，但供油策略的整体趋势仍然存在，即 “集群”情况的存在，这使得基于聚类的修正是可行的。

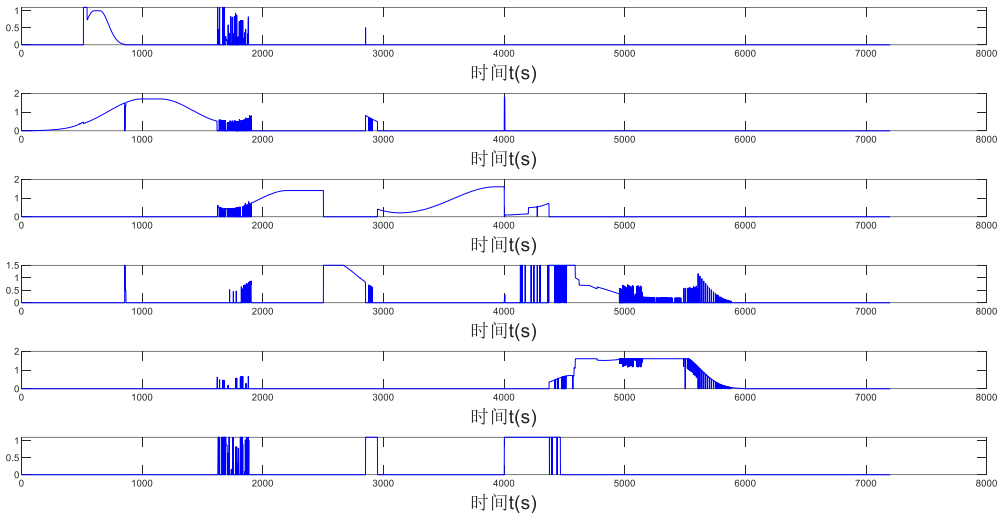

图 5- 3 修正前的供油策略

对供油策略进行分析聚类，修正得到的结果并重新多次迭代，其最终的六个油箱的供油曲线如图 5-4 所示，相比于之前的结论，其曲线变得更加光滑合理。

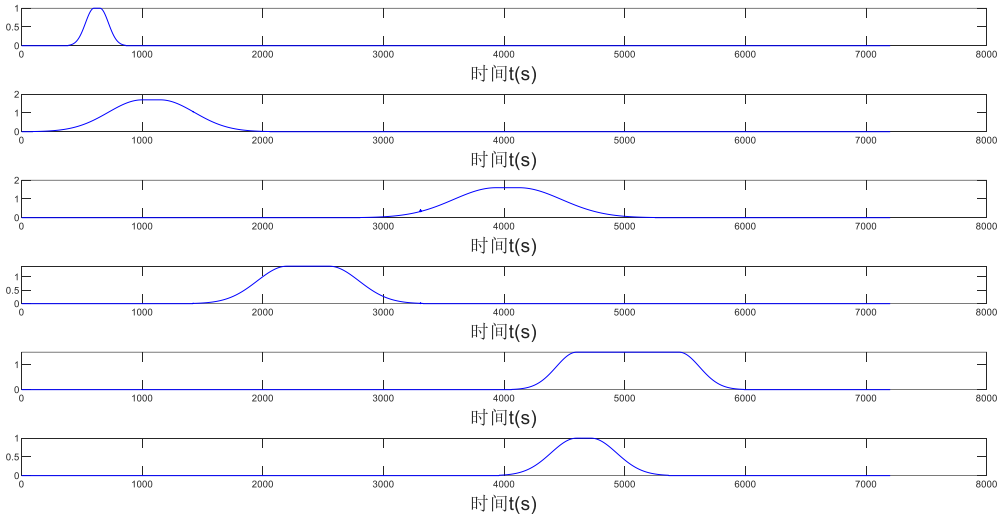

图 5- 4 修正后的供油策略

我们将 6 个油箱的供油速度（每秒一组）绘在一张图上，如图 5-5 所示。

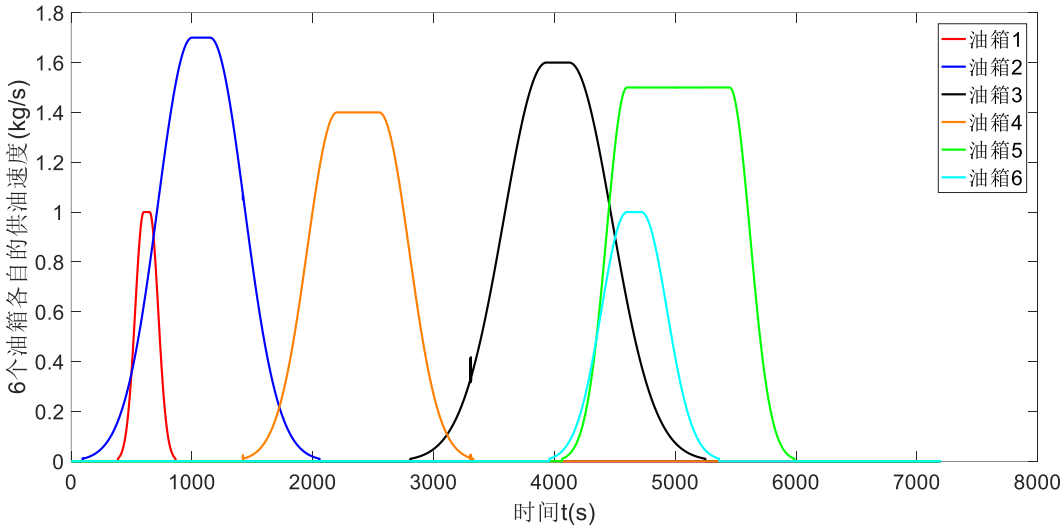

图 5-5 问题二 6 个油箱各自的供油曲线

根据上述结果，我们可以得到六个油箱的供油策略和选择方案。其中 1 号油箱在[387, 864]时间段供油，2 号油箱在[95, 2056]时间段供油，3 号油箱在[2802, 5251]时间段供油，4 号油箱在[1422, 3328]时间段供油，5 号油箱在[3959, 5987]时间段供油，6 号油箱在[3959, 5365]时间段供油。

将 4 个主油箱的供油速度相加，得到了 4 个主油箱的总供油速度（每秒一组），供油曲线如图 5-6 所示。与如图 5-7 所示的计划耗油曲线相比，二者基本可以重合。通过计算，4 个主油箱的总供油量为 6441.746226 千克，而发动机实际共耗油 6441.524212 千克，说明根据 5.3.1 所述的算法得到的供油策略，在保证实际质心与理想质心的偏移距离极小的前提下，导致的燃油浪费极少，浪费总量仅为 0.222014 千克。

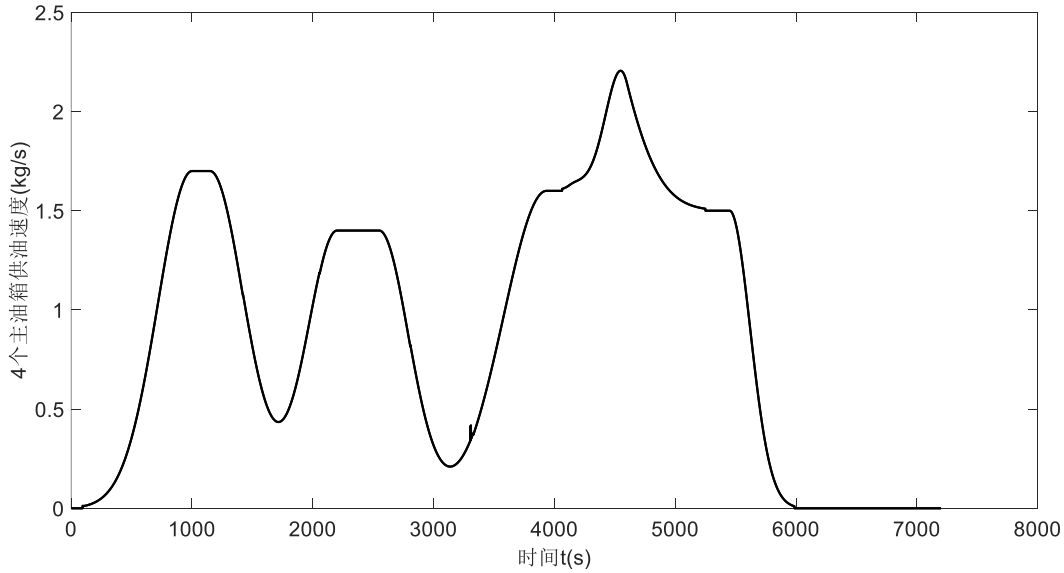

图 5-6 问题二 4 个主油箱的总供油速度曲线

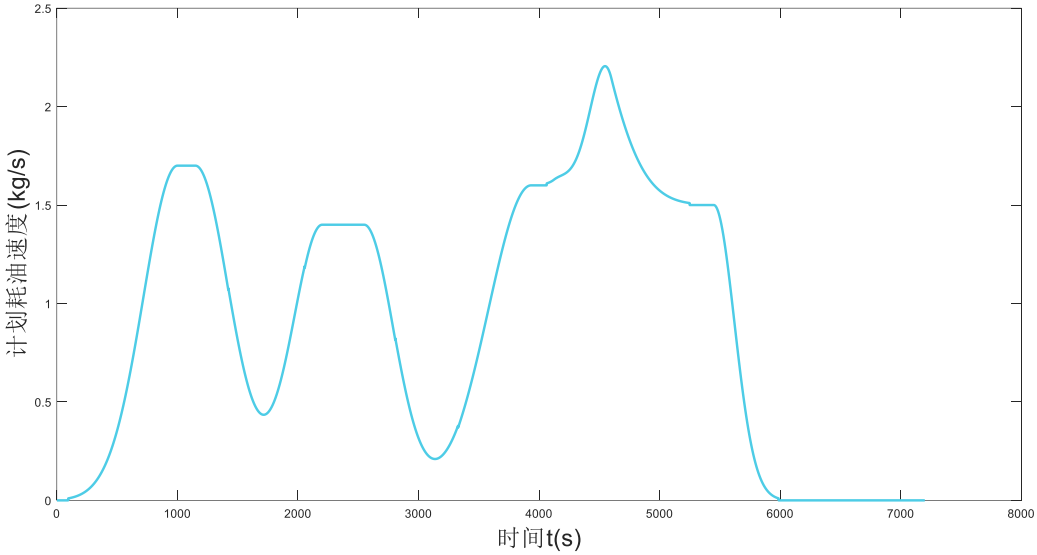

图 5-7 问题 2 计划耗油速度

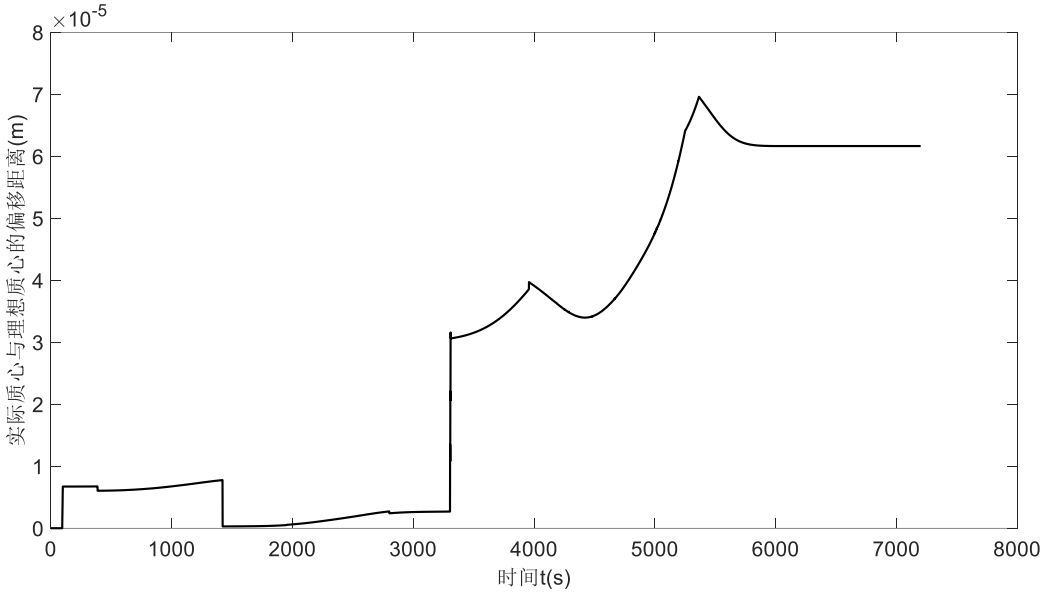

图 5- 8 实际质心与理想质心的欧氏距离

计算每一秒钟飞行器的实际质心与理想质心的欧氏距离，作出曲线如图 5-8 所示。实际质心与理想质心的欧氏距离最大值为 6.959780e-05 米，这一最大距离产生在第 5365 秒。可以认定，5.3.1 所述的算法求出了一种满足实际质心与理想质心的欧氏距离最大值很小的供油策略，完成了预期目标，且浪费的燃油极少。

为了进一步说明上述结论，我们设置不允许燃油浪费，即在 5.2 所述模型的约束 2 中，将 α 设定为 1，得到了一种新的供油策略。使用新的供油策略计算实际质心与理想质心的欧氏距离，得到的曲线如图 5-9 所示。

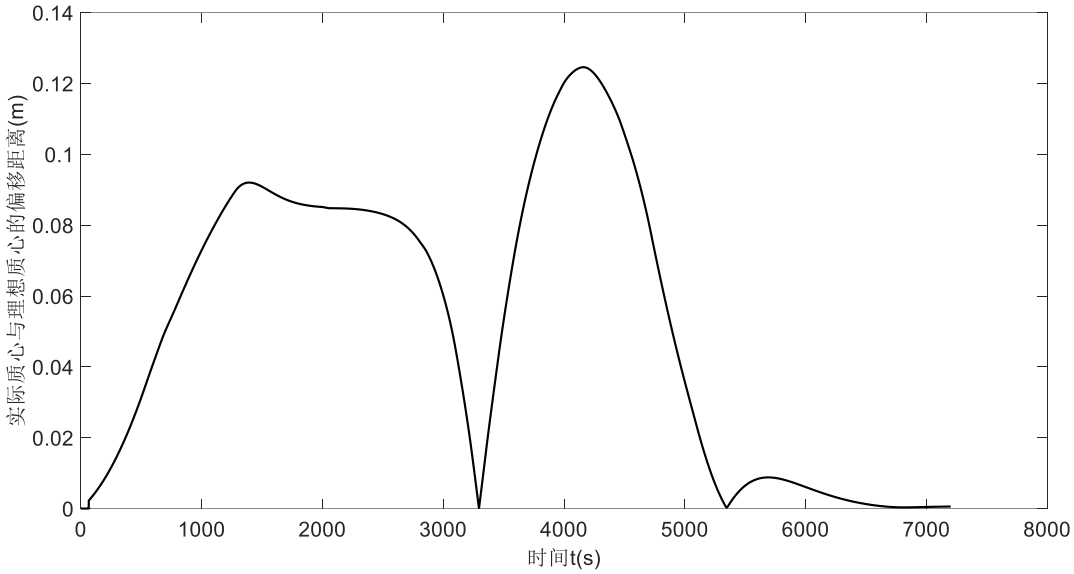

图 5- 9 不允许浪费燃油得到的实际质心与理想质心的欧氏距离

不允许浪费燃油得到的实际质心与理想质心的欧氏距离最大值为 0.1246 米，这一最大距离产生在第 4159 秒。这一最大距离相较于允许浪费时的情况，增大了约 1790 倍。

综上，使用所建立的模型和求解算法得到了一种供油策略，完成问题 2 所要求的任务后，4 个主油箱的总供油量为 6441.746226 千克，浪费燃油仅 0.222014 千克，实际质心与理想质心的欧氏距离最大值为 6.959780e-05 米。

6. 问题 3 模型的建立与求解

6.1 问题描述与分析

问题 3 相比于问题 2，同样是要求基于飞行器一次平飞任务中的耗油速度数据和理想质心数据规划飞行器在每个时刻每个油箱的供油速度，使得飞行器每一时刻的质心位置 $\vec{c}_1(t)$ 与理想质心位置 $\vec{c}_2(t)$ 的欧氏距离的最大值达到最小。不同点在于，问题 3 给定了任务结束时 6 个油箱剩余燃油总量的最小值，需要规划出每个油箱中燃油的初始量。

据此，我们将问题的解决分为两个方面，第一阶段确定任务开始时 6 个油箱的油量，第二阶段规划每个时间 6 个油箱的供油策略。

6.2 模型建立

第一阶段模型建立：

相比于第二题的求解，第三题需要求解所有油箱的初始油量。在允许浪费的情况下，不仅要求解初始油总量，也要求解各个油箱油量分配，该问题的解空间巨大，难以在短时间内求解。因此我们从最终状态反推。

第一阶段的目标是找出任务开始时，所剩燃油在 6 个油箱中的分布情况，即已知剩余燃油总量 $m(7200)$，将其分布在 6 个油箱中，使得质心偏离最终的理想质心最小。据此，我们可以确定目标函数与约束条件。

（1）目标函数

$$\begin{aligned}\min f &= \|\vec{c}_1(7200)-\vec{c}_2(7200)\|_2 \\ &= \sqrt{(x_{real}(7200)-x_{ideal}(7200))^2+(y_{real}(7200)-y_{ideal}(7200))^2+(z_{real}(7200)-z_{ideal}(7200))^2}\end{aligned} \tag{6-1}$$

其中 $(x_{ideal}(7200), y_{ideal}(7200), z_{ideal}(7200))$ 表示任务结束时飞行器在飞行器坐标系下的理想质心坐标，$(x_{real}(7200), y_{real}(7200), z_{real}(7200))$ 表示任务结束时飞行器在飞行器坐标系下的实际质心坐标，可以用各个油箱的质量和质心坐标求出：

$$\begin{cases} x_{real}(7200)=\dfrac{\sum_{i=1}^{6} m_i(7200)x_i(7200)}{M+\sum_{i=1}^{6} m_i(7200)x_i(7200)} \\ y_{real}(7200)=\dfrac{\sum_{i=1}^{6} m_i(7200)y_i(7200)}{M+\sum_{i=1}^{6} m_i(7200)y_i(7200)} \\ z_{real}(7200)=\dfrac{\sum_{i=1}^{6} m_i(7200)z_i(7200)}{M+\sum_{i=1}^{6} m_i(7200)z_i(7200)} \end{cases} \tag{6-2}$$

其中，$m_i(7200)$ 表示油箱 i 在任务结束时的所载燃油质量，$(x_i(7200), y_i(7200), z_i(7200))$ 表示油箱 i 在任务结束时在飞行器坐标系下的坐标，可以根据 $m_i(7200)$ 和空油箱的中心坐标求出。

综上，我们建立了实际质心与理想质心的欧氏距离关于 6 个油箱初始载油质量的目标函数。

（2）约束条件

约束 1：与问题 2 中的约束 3 类似，问题 3 也需要满足油箱容量约束：由于 6 个油箱的容量限制，每个油箱中燃油体积不能超过其能容纳的燃油质量上限 $m_{i\max}$：

$$m_{i\max} = a_i b_i c_i \times 850,\ \ i = 1,2,3,4,5,6 \tag{6-3}$$

与问题 2 不同的是，本问题中我们需要考虑 6 个油箱的容量上限：

$$m_{i\_start} \le m_{i\max},\ \ i = 1,2,3,4,5,6 \tag{6-4}$$

满足上述条件后，同时也必定就满足了 6 个油箱总容量的约束。

约束 2：耗油总量及剩余油量限制：本问题中，限制了任务结束时 6 个油箱内燃油剩余总量不低于 $m(7200)_{\min} = 850$。

综上，我们建立了第一阶段的数学模型。

第二阶段模型建立：

第二阶段的模型建立与问题 2 模型建立相同。

6.3 模型实现与求解

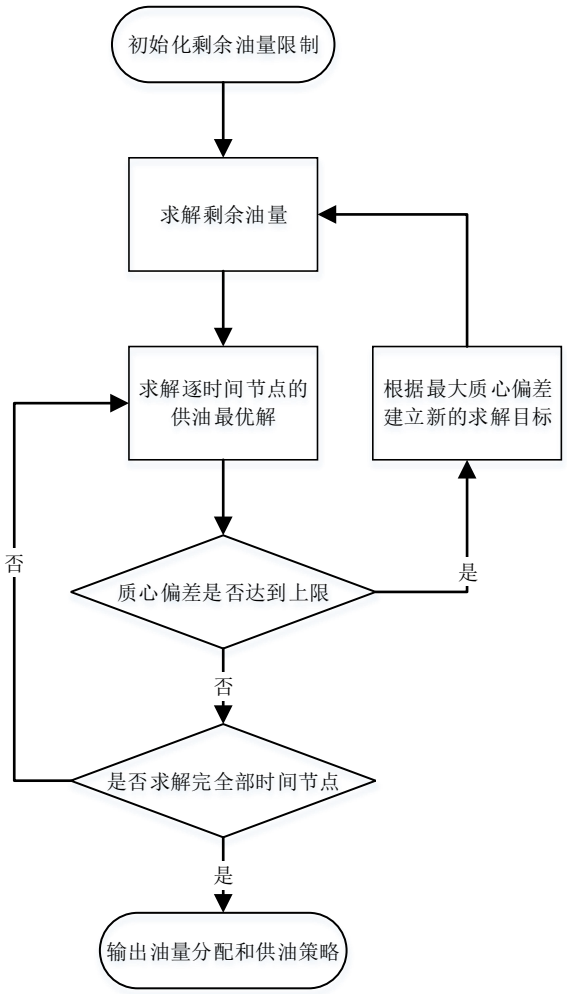

图 6- 1 问题 3 算法流程图

6.3.1 算法设计

相比于第二题，第三题需要确定初始油量，但这带来巨大的解空间和计算开销。因此

我们从最终状态开始反推飞行过程，直到得到初始状态的油量分配和供油策略。相应地，我们需要将问题拆解为两部分，第一阶段是求解最终剩余油量，第二阶段是根据最终油量反向推导飞行过程。

第一阶段我们求解剩余油量，其剩余油量的数学模型如 6.2 表示，我们利用 LINGO 进行求解。但最终状态的剩余油量的约束太少，有大量的可行解存在，因此我们采用贪心搜索的方案进行搜索，其流程如图 6-1 所示。

首先我们初始化剩余油量限制，即飞行最终阶段实际质心和理想质心偏差最小，由此可以得到剩余油量的初始解。再求解逐时间节点的供油最优解，该问题的求解与问题 2 模型求解类似，不同的是原本的油量消耗的方向取反，即变为油量供给。在逐时间节点的求解中，每一个时间节点都会计算当前实际质心与理想质心的偏差，如果超过偏差上限则不再继续计算，而根据当前节点的数据重新建立求解目标。以 t 时刻质心偏差超过上限为例，根据 t 之前时刻得到的供油策略，可以得到 t 时刻的油箱油量为

$$m_i(t) = m_i(7200) + \sum_{i=t}^{7200} v_i(t)\Delta t \tag{6-5}$$

根据 t 时刻的油箱油量，可以得到 t 时刻的实际质心位置与理想质心的偏差，对两者同时进行优化，即目标函数更新为

$$\min\ f = \max(\|\vec{c}_1(7200) - \vec{c}_2(7200)\|_2, \|\vec{c}_1(t) - \vec{c}_2(t)\|_2) \tag{6-6}$$

根据新的目标函数得到剩余油量的更优解，重新求解逐时间节点的供油最优解，直到求解完所有时间节点，并且所有质心偏差均满足偏差上限，此处我们的偏差上限设置为 0.2 米。由此我们得到了初步的供油方案，剩余油量和初始油量，进入第二阶段。

需要注意的时，为了简化计算流程，我们没有使用的与问题 2 类似的修正，因此得到的供油方案并不可靠，需要在第二阶段进行修正。同时，没有修正会带来一定的误差，导致剩余油量求解无法达到最优。但模型本身存在一定的容错性，因此带来可以容忍的误差，并且求解速度有了巨大的提升。

第二阶段与问题 2 类似，在得知了初始油量和初步的供油方案后，正向推导，进行修正和迭代直到满足问题 2 中的约束 5，也可以通过剩余油量和初步的供油方案，反向推导，进行修正迭代直到满足问题 2 中的约束 5。两者方案类似，此处我们用正向推导的方式进行求解，得到类似的剩余油量，以进行双重核验确保结果的准确性。由此我们得到最终的供油方案和初始油量。

6.3.2 问题 3 求解结果与分析

根据上述所述的算法流程，我们得到了 6 个油箱的供油速度（每秒一组），各自的供油曲线如图 6-2 所示，从上到下分别表示 1-6 油箱的供油曲线。

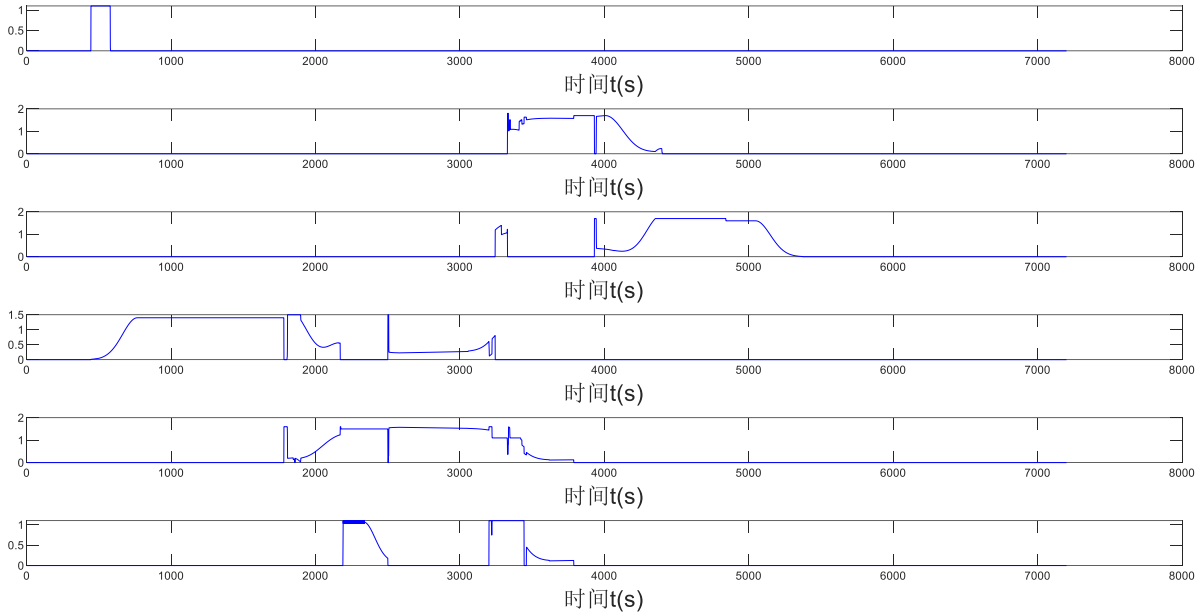

图 6- 2 问题 3 6 个油箱的耗油曲线

从上图中可以得出供油策略，从上往下依次是油箱 1~6 的供油曲线。可以看出油箱 1 主要在 600-800 时间输出，油箱 2 主要集中在 3300-4500 输出，油箱 3 主要在 3800-5200 输出，油箱 4 的供油时长较长，从 600-2200 及 2500-3300 是主要的输出时间段，油箱 5 的供油时间段主要为 1800-3800，油箱 6 的主要输出时间段为 2200-2500,3200-3800 两个时间段。

将 4 个主油箱的供油速度相加，得到了 4 个主油箱的总供油速度（每秒一组），供油曲线如下图所示。

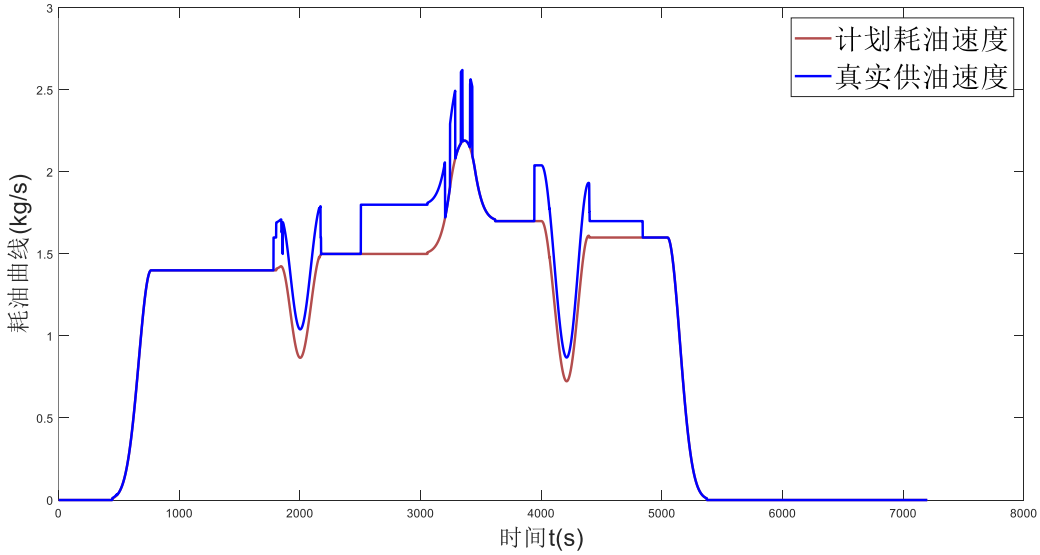

图 6- 3 实际供油曲线与计划耗油曲线对比图

通过计算，4 个主油箱的总供油量为 7298.005094 千克，而发动机实际共耗 6805.174669 千克。发动机油量有浪费，供油与消耗之比大约为 1.072。油箱 1-6 初始化的油量为(0.173535, 1.428274, 2.376, 2.652, 1.835641, 1.2)立方米。飞行器质心与理想质心距离的最大值为

0.119036 米，出现在第 5372 秒。其飞行器质心与理想质心的偏差曲线如图 6-4 所示。

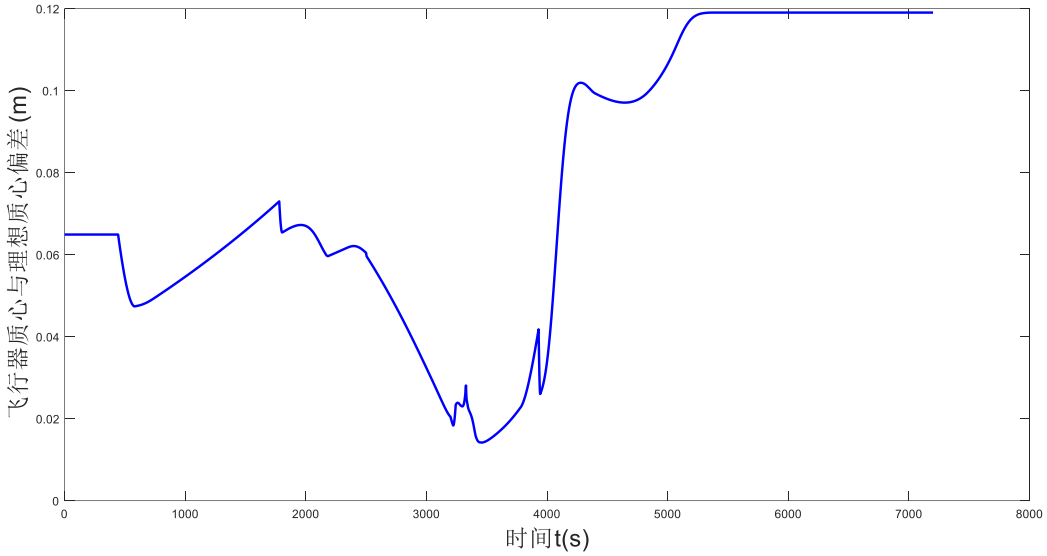

图 6- 4 飞行器质心与理想质心偏差示意图

可以看出，虽然 5000 秒之后的计划供油速度和实际供油速度均下降，但实际质心和理想质心偏差难以恢复，其原因在于此时大部分燃油均已被消耗，而 2-5 号油箱不能进行燃油相互传递使得质心难以被平衡，最终导致质心偏差居高不下。

7. 问题 4 模型的建立与求解

7.1 问题 4 描述与分析

相较于问题 2，问题 4 增加了飞行器在执行任务过程中的俯仰角的变化；相较于问题 3，问题 4 给定了 6 个油箱中油量的初始值。此外，问题 4 相较于问题 2 和问题 3 不再给定理想质心，改为要求任务执行过程中的实际质心与飞行器不载油时的质心的欧氏距离最小：

$$\min \max \| \vec{c}_1(t) - \vec{c}_0 \|_2 \tag{7-1}$$

综上，问题 4 的目标是根据飞行器的耗油速度和俯仰角变化，规划 6 个油箱的供油策略。

7.2 问题 4 模型建立

问题 4 的数学模型基本与问题 2 一致，但由于引入了俯仰角的变化，我们基于问题 1 的质心求解模型做了简化。

根据问题 1 中推导得到的计算质心方法及公式，在进行问题四中的迭代计算中，运算量急剧增大。相比于问题 2 及问题 3，问题四中引入俯仰角，所以质心计算公式从线性关系的计算中变成了分段式非线性计算，计算量变大，使得迭代时间变长。因此在保证质心偏移的相对正确性的基础上，为了减少计算量，可以对质心的求解进行简化。同时根据问题 1 中得到的质心偏移曲线中，发现系统整体质心的偏移量主要由 x 轴方向的偏移引起。

验证问题 1 系统整体质心的偏移量主要由 x 轴方向的偏移引起的推测。因此将问题 1 数据进行再次测试，将问题 1 中的俯仰角变成 0。得到新的质心偏移值，然后将原 x 轴质心坐标数据代替新生成的不加俯仰角 x 轴质心坐标，形成两组质心坐标。两种情况下的质心偏移数据曲线如下图所示。

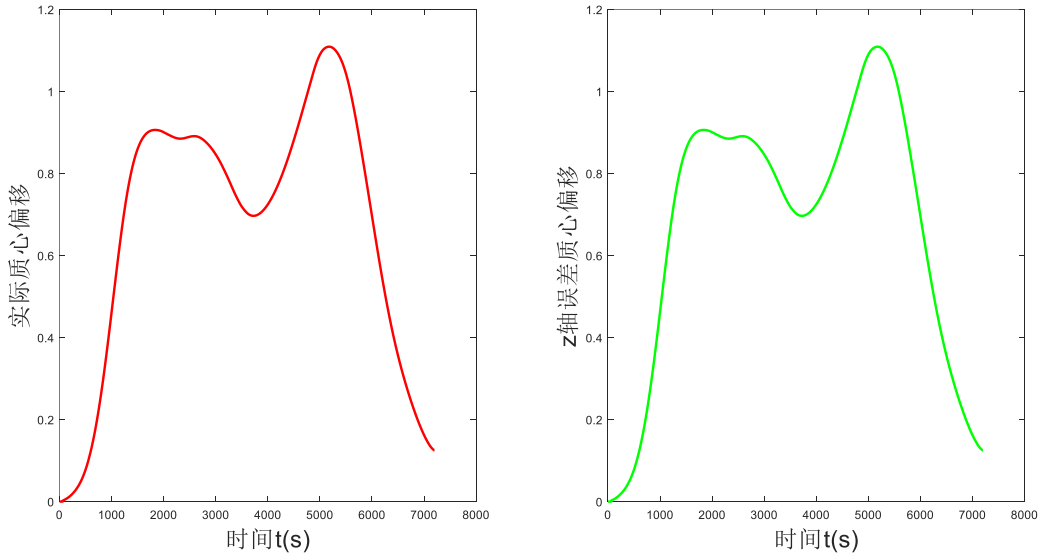

图 7- 1 z 轴误差导致质心偏移示意图

引入 z 轴误差后，两组曲线相同，同时放在同一图中会出现完全重合，说明 z 轴误差导致的质心偏移变化非常小。以下计算具体误差比例。

计算不加俯仰角时的质心位置相对原带有俯仰角的偏移比例。其由 z 轴数据引入的误

差数据如下图所示。

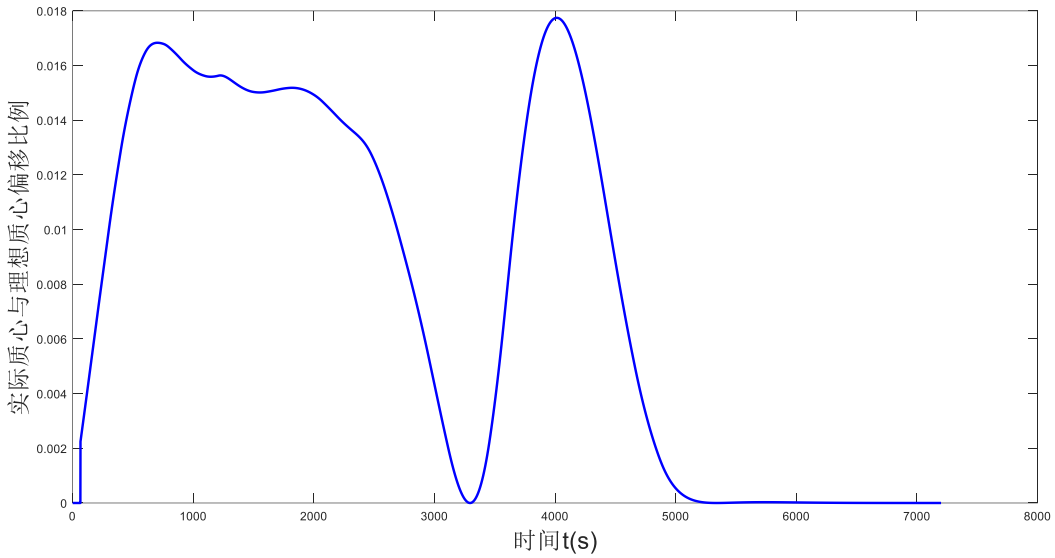

图 7- 2 z 轴误差导致实际质心与理想质心的偏移比例

由上图可以得出，由 z 轴数据引入的误差所占比最大为 1.8%。我们认为 z 轴数据引入的误差可以接受，因此在问题 4 中计算整体的质心模型时，可以只计算 x 轴向上的俯仰角变化引入的 x 轴质心偏移。Z 轴上质心数据按平飞时计算，采用此模型可以大幅减少计算质心所需计算量，加快迭代速度，同时误差在可接受范围内。

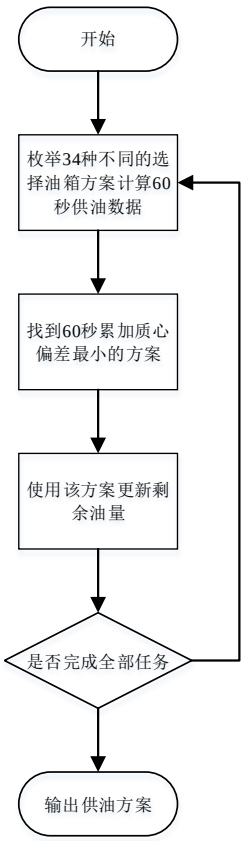

图 7- 3 问题 4 的求解算法流程图

7.3 问题 4 的模型实现与算法设计

7.3.1 算法设计

问题 4 的模型与问题 2 基本一致，但由于俯仰角的变化导致了求解质心坐标的过程中包含大量的分支，而使用 Lingo 软件求解包含大量分支的函数繁琐且复杂，因此，我们选择使用 Matlab 软件实现模型。

相较于问题 2，我们改变了求解过程，不再使用（找出最优解方案->根据约束 5 修正方案）的策略。我们将 60 秒时间划为一组，强制在 60 秒之内使用相同的油箱。根据供油油箱数约束，油箱使用方案共有 34 种，计算公式为：

$$(C_2^0 + C_2^1 + C_2^2)C_4^1 + (C_2^0 + C_2^1)C_4^2 = 34 \tag{7-2}$$

算法中，针对每一组 60 秒，我们求解采用不同方案的 60 秒最优供油策略，计算每种方案下 60 秒的质心偏差和，选择质心偏差和最小的方案作为该 60 秒的最终供油策略。依次循环，直到针对每组 60 秒都得到供油方案，即完成全部飞行任务。算法具体的流程图如图 7-3 所示。

求解过程的伪代码如下：

问题 4 的模型实现算法

输入:

```
consume: 发动机耗油数据
angle: 飞行器俯仰角
```

过程:

```
定义初始约束条件: 等式线性约束条件{Aeq, Beq}、不等式线性约束条件{A, B}、变量上下界{LB, UB}、搜寻起点{v0}
定义初始油箱油量 last
定义优化条件 options
for j=1 to 7200 %遍历所有时间, 步长为 60
    for i=1 to 34 %枚举 34 种方案
        更新剩余油量 lasttmp = last
        for k=1 to 60 %遍历 60 秒时长
            更新不等式约束条件边界 B
            fmincon(@(v)ques4sol(v,lasttmp,angle),v0,A,B,Aeq,Beq,LB,UB,'',
options);
            记录规划结果vt
            更新lasttmp
            distmp += dis; %累加当前方案的质心偏差
            更新变量v的上界UB
        end for
        dissum(i) = distmp; %记录当前方案的累计质心偏差
    end for
    q = min(dissum); %找到累计质心偏差最小方案
    fv(j:j+59,:) = vtmp(q,:,:); %将偏差最小的供油方案作为整体供油方案
```

```
    更新 last
    更新变量 v 的上界 UB
end for
```
输出： fv：问题 4 供油策略

7.3.2 问题 4 的求解结果与分析

根据 7.3.1 节所述的算法流程，我们得到了 6 个油箱的供油速度（每秒一组），各自的供油曲线如图 7-4 所示。

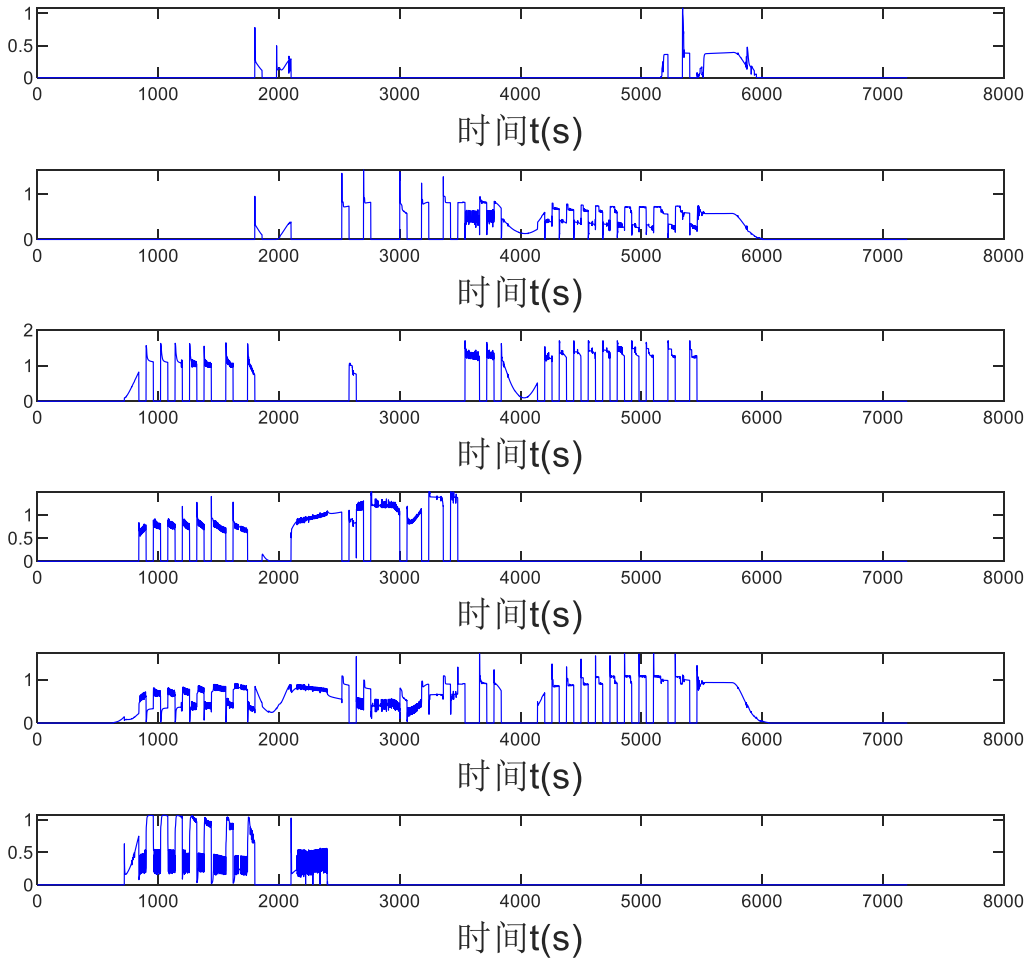

图 7-4　问题四 6 个油箱各自的供油曲线

将 4 个主油箱的供油速度相加，得到了 4 个主油箱的总供油速度（每秒一组），供油曲线如图 7-5 所示。与图中的计划耗油曲线相比，二者在大部分区域基本重合。通过计算，4 个主油箱的总供油量为 7565.579365 千克，而发动机实际共耗油 7035.545163 千克，说明根据 7.3.1 所述的算法得到的供油策略，在保证实际质心与理想质心的偏移距离很小的前提下，实际耗油约为计划耗油的 1.0753 倍。

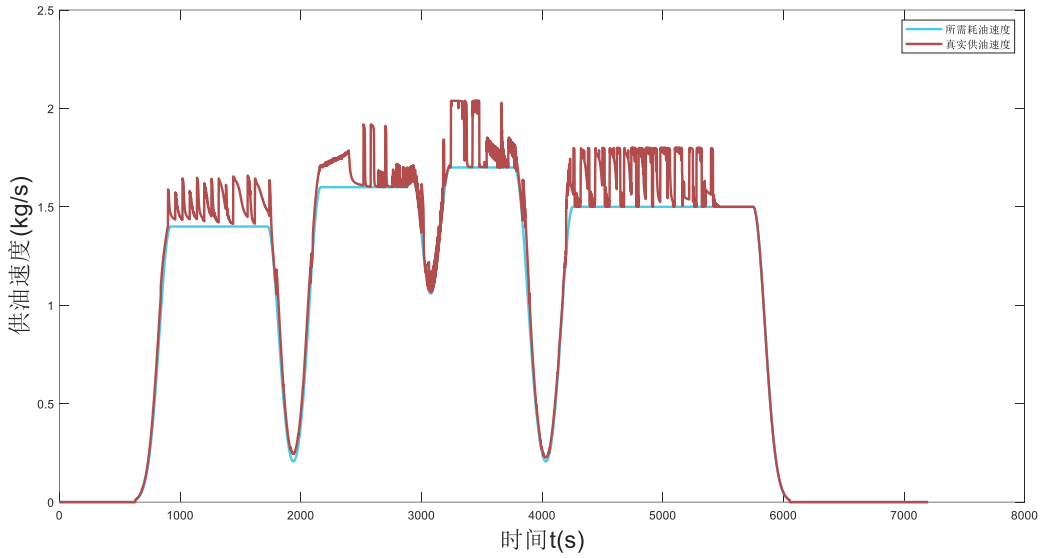

图 7-5 问题四 4 个主油箱的总供油速度曲线和计划耗油曲线

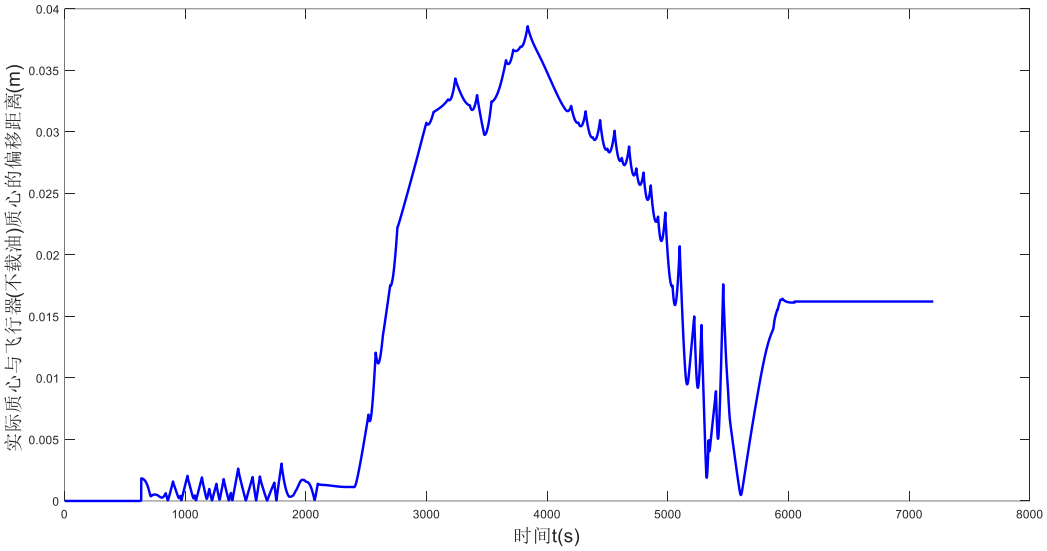

图 7-6 实际质心与飞行器（不载油）质心的欧氏距离

计算每一秒钟飞行器的实际质心与理想质心的欧氏距离，作出曲线如图 7-6 所示。实际质心与理想质心的欧氏距离最大值为 0.038594 米，这一最大距离产生在第 3839 秒。可以认定，7.3.1 所述的算法求出了一种满足实际质心与飞行器（不载油）质心的欧氏距离最大值很小的供油策略，完成了预期目标，且浪费的燃油很少。

综上，使用所建立的模型和求解算法得到了一种供油策略，完成问题 4 所要求的任务后，4 个主油箱的总供油量为 7565.579365 千克，为计划耗油的 1.0753 倍，实际质心与理想质心的欧氏距离最大值为 0.038594 米。

8.模型的总结与评价

8.1 模型的优点

1) 在问题 1 中，我们对飞行状态下的各个油箱质心进行建模，利用数学几何知识对三种状态下油量的质心计算进行求解，根据给定的油量和无人机仰角，能快速准确地求解出相对于无人机坐标系下的质心位置。

2) 问题 2 是根据飞行过程求解供油策略，为了最小化飞行过程中最大质心偏移，全局求解的思路因大量的变量难以求解。我们利用贪心的思路对问题进行了简化，相比于基于整个飞行过程的优化求解，逐时间节点的优化求解能极大的加快求解答速度，并根据数据分布规律，参考聚类算法对数据进行修正，能准确快速地得到满足所有约束的供油方案。

3) 问题 3 需要同时确定油量分配和供油策略，我们引入了基于贪心搜索的算法求解剩余油量，相比于其他启发式算法能更快地得到一个最优解。

4) 问题 4 在理想质心不变且恒为原点的情况下，进一步简化了供油策略的求解。将整个飞行过程按照 60 秒的时间进行划分，极大的加速了求解速度，且求解不需要进行进一步的修正，能够更快速准确地得到满足所有约束的供油方案。问题 4 还对飞行器存在仰角的情况下，简化质心求解过程，在误差允许的情况下能大幅加速供油策略的求解。

8.2 模型的改进

1) 基于数据分布规律的供油方案修正策略依赖数据分布规律，在极端情况下可能无法得到合理的修正。

2) 基于贪心搜索求解剩余油量的算法对算法起始点的敏感度高，影响最终收敛值。较差的起始点可能使得算法收敛到局部最优。

3) 问题 4 提出的，对飞行器存在仰角的情况下，质心求解的简化过程仅在仰角较小的情况下有效，随着仰角的增大误差增大而变得难以接受。

8.3 展望

1) 参考其他启发式算法，改进加快基于贪心搜索求解剩余油量的算法速度，提高算法的稳定性，使其更容易收敛到全局最优点。

2) 对于复杂情况下的质心模型求解可以进行相应的优化，在可接受误差下减少贪心搜索的计算量。

3) 本题没有引入左右偏转的情况，问题考虑略简单，离真实情况还有很大的差距。在今后的工作中应当建立更完整的数学模型以得到更具有现实意义的结论。

参考文献

[1]曾关辉,王继亮.受油飞机重量重心控制技术研究[J].河南科技,2020(17):41-43.
[2]吕亚国. 飞机燃油系统计算研究[D].西北工业大学,2006.
[3]张宇庆.Cessna 172R 飞机油箱燃油消耗不平衡浅析[J].内燃机与配件,2019(23):155-157.
[4] 邹新龙, 石丹, 刘茂, 毕军建, 谭志良,ZOU Xin-long, SHI Dan, LIU Mao, BI Jun-jian, TAN Zhi-liang- 《环境技术》2014 年 z2 期

附录

问题一：

[Centroid solution:]

```
clc;
clear;
q=zeros(6,7);
watch=zeros(1,6);
[Consume]=xlsread('C:\Users\DELL\Desktop\新建文件夹\附件 2-问题 1 数据.xlsx',1,'B2:G7201');
Centroid_0=zeros(7200,3);

[angles]=xlsread('C:\Users\DELL\Desktop\新建文件夹\附件 2-问题 1 数据.xlsx',2,'B2:B7201');
centerOffset=[8.91304348 1.20652174 0.61669004;
              6.91304348  -1.39347826  0.21669004;
              -1.68695652 1.20652174   -0.28330996;
              3.11304348  0.60652174   -0.18330996;
              -5.28695652  -0.29347826  0.41669004;
              -2.08695652  -1.49347826  0.21669004];
  sizeOffset=[1.5  0.9  0.3;
              2.2  0.8  1.1;
              2.4  1.1  0.9;
              1.7  1.3  1.2;
              2.4  1.2  1;
              2.4  1  0.5];
remainingOil=[0.3 1.5 2.1 1.9 2.6 0.8];
%remainingOil=[0.173535 1.428274 2.376 2.652 1.835641 1.2];
for  i=1:7200
    for j=1:6
       remainingOil(j)=remainingOil(j)-Consume(i,j)/850;
    end
    remainingOil(2)=remainingOil(2)+(Consume(i,1)/850);
    remainingOil(5)=remainingOil(5)+(Consume(i,6)/850);

    tmpCentroid.x=0;
    tmpCentroid.y=0;
    tmpCentroid.z=0;
    tmpWeight=0;
    for j=1:6
        tmp.y=centerOffset(j,2);
        a=sizeOffset(j,1);
        c=sizeOffset(j,3);
        radian = angles(i) * pi /180;
```

```
        s = remainingOil(j) / sizeOffset(j,2);
        angle_t = abs(radian);
        if angle_t==0
            x=s/a;
            tmp.x=0;
            tmp.z=-c/2+x/2;
            q(j,1)=q(j,1)+1;
        elseif  angle_t<abs(atan(c / a))
            boundary1 = a * a * tan(angle_t) / 2;
            boundary2 = a * c - a * a * tan(angle_t) / 2;
            if s<=boundary1
                  x = sqrt(2 * s / tan(angle_t));
                  x1 = -a/2;
                  y1 = -c/2;
                  x2 = -a / 2;
                  y2 = x * tan(angle_t)-c/2;
                  x3 = x-a/2;
                  y3 = -c/2;
                  q(j,2)=q(j,2)+1;
                  tmp.x=(x1+x2+x3)/3;
                  tmp.z = (y1 + y2 + y3) / 3;
            elseif s <= boundary2
                 x = s / a + a * tan(angle_t) / 2;
                 x1 = -a / 2;
                 y1 = -c / 2;
                 x2 = a / 2;
                 y2 = -c / 2;
                 x3 = a / 2;
                 y3 = -c/2 + x-a*tan(angle_t);
                 x4 = -a / 2;
                 y4 = x - c / 2;
                 q(j,3)=q(j,3)+1;
                 input=[x1 y1;x2 y2;x3 y3;x4 y4];

                 [tmp.x,tmp.z]=PolygonCentroid(input);
            else
                 x = a-sqrt(2*(a*c-s)/tan(angle_t));
                 x1 = -a / 2;
                 y1 = -c / 2;
                 x2 = a / 2;
                 y2 = -c / 2;
                 x3 = a / 2;
                 y3 = c / 2 - (a-x)* tan(angle_t);
                 x4 = -a/2 +x;
```

```
                y4 =c/2;
                x5 = -a / 2;
                y5 = c / 2;
                input=[x1 y1;x2 y2;x3 y3;x4 y4;x5 y5];
                q(j,4)=q(j,4)+1;
                [tmp.x,tmp.z]=PolygonCentroid(input);
            end
        else
             boundary1 = c * c /( tan(angle_t) * 2);
             boundary2 = a * c - c * c /( tan(angle_t) * 2);
             if s<=boundary1
                  x = sqrt(2 * s / tan(angle_t));
                  x1 = -a/2;
                  y1 = -c/2;
                  x2 = -a / 2;
                  y2 = x * tan(angle_t)-c/2;
                  x3 = x-a/2;
                   y3 = -c/2;
                   q(j,5)=q(j,5)+1;
                   tmp.x=(x1+x2+x3)/3;
                   tmp.z = (y1 + y2 + y3) / 3;
              elseif s <= boundary2
                  x = s / c + c/(tan(angle_t) * 2);
                  x1 = -a / 2;
                  y1 = -c / 2;
                  x2 = x-a / 2;
                  y2 = -c / 2;
                  x3 = -a/2 + x -c/tan(angle_t);
                  y3 = c/2 ;
                  x4 = -a / 2;
                  y4 = c / 2;
                  q(j,6)=q(j,6)+1;
                  input=[x1 y1;x2 y2;x3 y3;x4 y4];
                  [tmp.x,tmp.z]=PolygonCentroid(input);
              else
                  x = a-sqrt(2*(a*c-s)/tan(angle_t));
                  x1 = -a / 2;
                  y1 = -c / 2;
                  x2 = a / 2;
                  y2 = -c / 2;
                  x3 = a / 2;
                  y3 = c / 2 - (a-x)* tan(angle_t);
                  x4 = -a/2 +x;
                  y4 =c/2;
```

```
                x5 = -a / 2;
                y5 = c / 2;
                q(j,7)=q(j,7)+1;
                input=[x1 y1;x2 y2;x3 y3;x4 y4;x5 y5];
                [tmp.x,tmp.z]=PolygonCentroid(input);
             end
        end
        if angles(i)<0
            tmp.x=-tmp.x;
            tmp.z=tmp.z;
        end
        tmp.x=centerOffset(j,1)+tmp.x;
        tmp.z=centerOffset(j,3)+tmp.z;
        if i==66
            watch(j)=tmp.z;
        end
        tmpCentroid.x=tmpCentroid.x+tmp.x*remainingOil(j)*850;
        tmpCentroid.y=tmpCentroid.y+tmp.y*remainingOil(j)*850;
        tmpCentroid.z=tmpCentroid.z+tmp.z*remainingOil(j)*850;
        tmpWeight=tmpWeight+remainingOil(j)*850;
    end
    Centroid_0(i,1)=tmpCentroid.x/(tmpWeight+3000);
    Centroid_0(i,2)=tmpCentroid.y/(tmpWeight+3000);
    Centroid_0(i,3)=tmpCentroid.z/(tmpWeight+3000);
end
%plot(1:7200,Centroid_0(:,1),'r','LineWidth',2);
%hold on
%plot(1:7200,Centroid_0(:,2),'b','LineWidth',2);
%hold on
%plot(1:7200,Centroid_0(:,3),'g','LineWidth',2);
%legend('x 方向偏移','y 方向偏移','z 方向偏移');
%plot(1:7200,Absolute_no,'-k','LineWidth',2);
%set(gca,'FontSize',20);
%axis([0,8000,0,8e-5])
%xlabel('时间 t(s)','fontsize',20);
%ylabel('实际质心与理想质心的偏移距离(m)','fontsize',20)
%[m,index]=max(Centroid_0);
res=zeros(7200,1);
for i=1:7200
    res(i)=norm(Centroid_0(i,:));
end
[m,index]=max(res);
an=0;
for i=1:7200
```

```
171.     an=Consume(i,2)+Consume(i,3)+Consume(i,4)+Consume(i,5)+an;
172. end
173.  %plot(1:7200,Consume,'-b','LineWidth',2);
174.  %xlabel('时间 t(s)','fontsize',16);
175.  %ylabel('各个供油曲线','fontsize',16)
176.  figure(6)           % define figure
177.  subplot(2,3,1);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几幅
    图片
178. plot(1:7200,Consume(:,1),'b');
179.  hold on
180.  xlabel('时间 t(s)','fontsize',16);
181.  ylabel('油箱 1 供油速度(kg/s)','fontsize',16);
182. subplot(2,3,2);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几幅
    图片
183.  plot(1:7200,Consume(:,2),'b');
184. hold on
185.  xlabel('时间 t(s)','fontsize',16);
186.  ylabel('油箱 2 供油速度(kg/s)','fontsize',16);
187.
188.  subplot(2,3,3);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几幅
    图片
189.  plot(1:7200,Consume(:,3),'b');
190.  hold on
191.  xlabel('时间 t(s)','fontsize',16);
192.  ylabel('油箱 3 供油速度(kg/s)','fontsize',16);
193.
194.  subplot(2,3,4);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几幅
    图片
195.  plot(1:7200,Consume(:,4),'b');
196.  hold on
197.  xlabel('时间 t(s)','fontsize',16);
198.  ylabel('油箱 4 供油速度(kg/s)','fontsize',16);
199. %
200. subplot(2,3,5);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几幅
    图片
201.  plot(1:7200,Consume(:,5),'b');
202.  hold on
203.  xlabel('时间 t(s)','fontsize',16);
204.  ylabel('油箱 5 供油速度(kg/s)','fontsize',16);
205.  % subplot(2,3,6);      % subplot(x,y,n)x 表示显示的行数，y 表示列数，n 表示第几
    幅图片
206. % plot(1:7200,Consume(:,6),'b');
207. % hold on
208. % xlabel('时间 t(s)','fontsize',16);
```

```
% ylabel('油箱 6 供油速度(kg/s)','fontsize',16);
function [x,z]=PolygonCentroid(input)
    [count,~]=size(input);
    totalArea =0;
    totalTriangleCentroid.x=0;
    totalTriangleCentroid.z=0;
    for i=2:count-1
        [temp,tempCentroid.x,tempCentroid.z]=TriangleCentroid(input(1,:),input(i,:),input(i+1,:));
        totalTriangleCentroid.x=tempCentroid.x*temp+totalTriangleCentroid.x;
        totalTriangleCentroid.z=tempCentroid.z*temp+totalTriangleCentroid.z;
        totalArea=totalArea+temp;
    end
    x=totalTriangleCentroid.x/totalArea;
    z=totalTriangleCentroid.z/totalArea;
end
function [temp,x,z]=TriangleCentroid(input1,input2,input3)
    x1=input2(1,1)-input1(1,1);
    y1=input2(1,2)-input1(1,2);
    x2=input3(1,1)-input1(1,1);
    y2=input3(1,2)-input1(1,2);
    temp = (x1 * y2 - x2 * y1) / 2;
    x=(input1(1,1)+input2(1,1)+input3(1,1))/3;
    z=(input1(1,2)+input2(1,2)+input3(1,2))/3;
end
```

[C++代码]

```
#include<iostream>
#include<vector>
#include<math.h>
#include<unordered_map>
#include <fstream>
#include <cassert>
#include<string>

using namespace std;

#define PI 3.141592653589793
 #define Density (double)850
 #define netWeight (double)3000

```

```cpp
struct Vec2d
{
    double x;
    double y;
    Vec2d(double a, double b) { x = a; y = b; };
    Vec2d() { x = 0; y = 0; };
    Vec2d operator-(const Vec2d& b)
    {
        Vec2d vec2d(0, 0);
        vec2d.x = this->x - b.x;
        vec2d.y = this->y - b.y;
        return vec2d;
    }
};

//叉乘算三角形面积(有正负)，以及算三角形质心坐标
void  TriangleCentroid(Vec2d P1, Vec2d P2, Vec2d P3, double& area, Vec2d& centroid)
{
    Vec2d P12 = P2 - P1;
    Vec2d P13 = P3 - P1;

    double x1 = P12.x;
    double y1 = P12.y;

    double x2 = P13.x;
    double y2 = P13.y;


    //向量叉乘计算三角形面积
    area = (x1 * y2 - x2 * y1) / 2;

    //计算三角形质心
    centroid.x = (P1.x + P2.x + P3.x) / 3;
    centroid.y = (P1.y + P2.y + P3.y) / 3;

    return;
}

//计算质心坐标，多边形的顶点按顺序排列在数组中
bool PolygonCentroid(vector<Vec2d> Points, Vec2d& centroid)
{
    int count = Points.size();

```

```
58.     //如果不是多边形，不计算
59.     if (count < 3)
60.         return false;
61.
62.     //多边形总面积
63.     double totalArea = 0;
64.
65.     //n 边形划分成 n-2 个三角形，每个三角形质心坐标与该三角形面积乘积之和
66.     Vec2d totalTriangleCentroid(0, 0);
67.
68.     //按顶点顺序，1,2,3；1,3,4；1,4,5；如此构建三角形
69.     for (int i = 1; i < count - 1; i++)
70.     {
71.         double temp;
72.         Vec2d tempCentroid(0, 0);
73.
74.         //计算每个三角形的面积与质心坐标
75.         TriangleCentroid(Points[0], Points[i], Points[i + 1], temp, tempCe
    ntroid);
76.
77.         totalTriangleCentroid.x += tempCentroid.x * temp;
78.         totalTriangleCentroid.y += tempCentroid.y * temp;
79.
80.         totalArea += temp;
81.     }
82.
83.
84.     //计算质心坐标
85.     centroid.x = totalTriangleCentroid.x / totalArea;
86.     centroid.y = totalTriangleCentroid.y / totalArea;
87.
88.     return true;
89. }
90.
91. struct point
92. {
93.     double x;
94.     double y;
95.     double z;
96.     point(double a, double b, double c) { x = a; y = b; z = c; };
97.     point() { x = 0; y = 0; z = 0; };
98. };
99.
```

```
unordered_map<int, point> centerOffset = { {1,{8.91304348,1.20652174,0.61669004}},
                                           {2,{6.91304348,-1.39347826,0.21669004}},
                                           {3,{-1.68695652,1.20652174,-0.28330996}},
                                           {4,{3.11304348,0.60652174,-0.18330996}},
                                           {5,{-5.28695652,-0.29347826,0.41669004}},
                                           {6,{-2.08695652,-1.49347826,0.21669004}} };
unordered_map<int, point> sizeOffset = { {1,{1.5,0.9,0.3}},
                                         {2,{2.2,0.8,1.1}},
                                         {3,{2.4,1.1,0.9}},
                                         {4,{1.7,1.3,1.2}},
                                         {5,{2.4,1.2,1}},
                                         {6,{2.4,1,0.5}} };
unordered_map<int,double> remainingOil={{1,0.3},{2,1.5},{3,2.1},{4,1.9},{5,2.6},{6,0.8}};
void addOffset(int num, point& tmp, double angle) {
    tmp.x = centerOffset[num].x +tmp.x;
    tmp.z = centerOffset[num].z +tmp.z;
}
Vec2d coordinate(double angle, int num, double speed){
    double a = sizeOffset[num].x;
    double c = sizeOffset[num].z;
    double radian = angle * PI / (double)180;
    double s = remainingOil[num] / sizeOffset[num].y;
    double angle_t = fabs(radian);

    Vec2d Centroid;

    if(angle==0){
        double x=s/a;
        Centroid.x = 0;
        Centroid.y = -c / 2 + x / 2;
        return Centroid;
    }else if (angle_t < fabs(atan(c / a))) {
        double boundary1 = a * a * tan(angle_t) / 2;
        double boundary2 = a * c - a * a * tan(angle_t) / 2;
        if (s <= boundary1) {
            double x = pow(2 * s / tan(angle_t), 0.5);
            double x1 = -a/2;
```

```
137.             double y1 = -c/2;
138.             double x2 = -a / 2;
139.             double y2 = x * tan(angle_t)-c/2;
140.             double x3 = x-a/2;
141.             double y3 = -c/2;
142.
143.             Centroid.x = (x1 + x2 + x3) / 3;
144.             Centroid.y = (y1 + y2 + y3) / 3;
145.
146.         }
147.         else if (s <= boundary2) {
148.             double x = s / a + a * tan(angle_t) / 2;
149.             double x1 = -a / 2;
150.             double y1 = -c / 2;
151.             double x2 = a / 2;
152.             double y2 = -c / 2;
153.             double x3 = a / 2;
154.             double y3 = -c/2 + x-a*tan(angle_t);
155.             double x4 = -a / 2;
156.             double y4 = x - c / 2;
157.
158.             vector<Vec2d> Points = { {x1,y1},{x2,y2},{x3,y3},{x4,y4} };
159.             PolygonCentroid(Points, Centroid);
160.
161.         }
162.         else {
163.             double x = a-pow(2*(a*c-s)/tan(angle_t),0.5);
164.             double x1 = -a / 2;
165.             double y1 = -c / 2;
166.             double x2 = a / 2;
167.             double y2 = -c / 2;
168.             double x3 = a / 2;
169.             double y3 = c / 2 - (a-x)* tan(angle_t);
170.             double x4 = -a/2 +x;
171.             double y4 =c/2;
172.             double x5 = -a / 2;
173.             double y5 = c / 2;
174.
175.             vector<Vec2d> Points = { {x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,
    y5} };
176.             PolygonCentroid(Points, Centroid);
177.
178.         }
179.
```

```
180.     }else
181.     {
182.         double boundary1 = c * c /( tan(angle_t) * 2);
183.         double boundary2 = a * c - c * c /( tan(angle_t) * 2);
184.         if (s <= boundary1) {
185.             double x = pow(2 * s / tan(angle_t), 0.5);
186.             double x1 = -a/2;
187.             double y1 = -c/2;
188.             double x2 = -a / 2;
189.             double y2 = x * tan(angle_t)-c/2;
190.             double x3 = x-a/2;
191.             double y3 = -c/2;
192.
193.             Centroid.x = (x1 + x2 + x3) / 3;
194.             Centroid.y = (y1 + y2 + y3) / 3;
195.         }
196.         else if (s <= boundary2) {
197.             double x = s / c + c/(tan(angle_t) * 2);
198.             double x1 = -a / 2;
199.             double y1 = -c / 2;
200.             double x2 = x-a / 2;
201.             double y2 = -c / 2;
202.             double x3 = -a/2 + x -c/tan(angle_t);
203.             double y3 = c/2 ;
204.             double x4 = -a / 2;
205.             double y4 = c / 2;
206.
207.             vector<Vec2d> Points = { {x1,y1},{x2,y2},{x3,y3},{x4,y4} };
208.             PolygonCentroid(Points, Centroid);
209.
210.         }
211.         else {
212.             double x = a-pow(2*(a*c-s)/tan(angle_t),0.5);
213.             double x1 = -a / 2;
214.             double y1 = -c / 2;
215.             double x2 = a / 2;
216.             double y2 = -c / 2;
217.             double x3 = a / 2;
218.             double y3 = c / 2 - (a-x)* tan(angle_t);
219.             double x4 = -a/2 +x;
220.             double y4 =c/2;
221.             double x5 = -a / 2;
222.             double y5 = c / 2;
223.
```

```
            vector<Vec2d> Points = { {x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5} };
            PolygonCentroid(Points, Centroid);
        }
    }
    if(angle>0){
        return Centroid;
    }else
    {
        Centroid.x=-Centroid.x;
        Centroid.y=Centroid.y;
        return Centroid;
    }

}

int main(void) {
    vector<double> angles;
    ifstream infile;
    infile.open("D:\\math_data\\angle.txt");   //将文件流对象与文件连接起来
    assert(infile.is_open());   //若失败,则输出错误消息,并终止程序运行

    string s;
    while(getline(infile,s))
    {
        angles.push_back(atof(s.c_str()));
    }
    infile.close();             //关闭文件输入流
    unordered_map<int, vector<double>> Consume;
    for (size_t i = 1; i < 7; i++)
    {
        ifstream infile;
        infile.open("D:\\math_data\\data"+to_string(i)+".txt");   //将文件流对象与文件连接起来
        assert(infile.is_open());   //若失败,则输出错误消息,并终止程序运行

        string s;
        while(getline(infile,s))
        {
        Consume[i].push_back(atof(s.c_str()));
        }
        infile.close();             //关闭文件输入流
```

```
264.     }
265.
266.     vector<point> Centroid_s;
267.     for (int i = 0; i < angles.size(); i++)
268.     {
269.         //计算剩余油量
270.         for(int j=1;j<7;++j){
271.             remainingOil[j]-=(Consume[j][i]/Density);
272.         }
273.         remainingOil[2]=remainingOil[2]+(Consume[1][i]/Density);
274.         remainingOil[5]=remainingOil[5]+(Consume[6][i]/Density);
275.         //计算质心
276.         point tmpCentroid;
277.         double tmpWeight=0;
278.         for (size_t j = 1; j < 7; j++)
279.         {
280.             Vec2d centroid=coordinate(angles[i],j,Consume[j][i]);
281.             point tmp(centroid.x,centerOffset[j].y,centroid.y);
282.
283.             addOffset(j,tmp,angles[i]);
284.             /*if (j == 1) {
285.                 ofstream ofs;
286.                 ofs.open("D:\\math_data\\Centroid2.txt", ios::app);
287.                 ofs << tmp.x <<endl;
288.                 //ofs << tmp.y << "    ";
289.                 //ofs << tmp.z << endl;
290.
291.                 ofs.close();
292.             }*/
293.             vector<double> watch;
294.             watch.resize(6);
295.             if (i == 66) {
296.                 watch[j - 1] = tmp.z;
297.                 cout << tmp.z << endl;
298.             }
299.             tmpCentroid.x+=tmp.x*remainingOil[j]* Density;
300.             tmpCentroid.y+=tmp.y*remainingOil[j]* Density;
301.             tmpCentroid.z+=tmp.z*remainingOil[j]* Density;
302.             tmpWeight+=remainingOil[j]* Density;
303.         }
304.         tmpCentroid.x=tmpCentroid.x/(tmpWeight+netWeight);
305.         tmpCentroid.y=tmpCentroid.y/(tmpWeight+netWeight);
306.         tmpCentroid.z=tmpCentroid.z/(tmpWeight+netWeight);
307.         Centroid_s.push_back(tmpCentroid);
```

```
    }
    {
        ofstream ofs;
        ofs.open("D:\\math_data\\out0.txt", ios::out);
        for (size_t i = 0; i < Centroid_s.size(); i++)
        {
            ofs << Centroid_s[i].x<< endl;
        }
        ofs.close();

        ofs.open("D:\\math_data\\out1.txt", ios::out);
        for (size_t i = 0; i < Centroid_s.size(); i++)
        {
            ofs << Centroid_s[i].y << endl;
        }
        ofs.close();

        ofs.open("D:\\math_data\\out2.txt", ios::out);
        for (size_t i = 0; i < Centroid_s.size(); i++)
        {
            ofs << Centroid_s[i].z << endl;
        }
        ofs.close();
    }
    return 0;
}
```

问题二：
[求解逐时间节点供油最优解的 python 控制代码]

```
## for dt = 1 without constrain of 60s

import os
from glob import glob
import subprocess as sp
import sys
import openpyxl
import time
from tqdm import tqdm
class PowerShell:
    # from scapy
    def __init__(self, coding, ):
        cmd = [self._where('PowerShell.exe'),
               "-NoLogo", "-NonInteractive",  # Do not print headers
               "-Command", "-"]  # Listen commands from stdin
```

```
        startupinfo = sp.STARTUPINFO()
        startupinfo.dwFlags |= sp.STARTF_USESHOWWINDOW
        self.popen = sp.Popen(cmd, stdout=sp.PIPE, stdin=sp.PIPE, stderr=sp.STDOUT, sta
rtupinfo=startupinfo)
        self.coding = coding

    def __enter__(self):
        return self

    def __exit__(self, a, b, c):
        self.popen.kill()

    def run(self, cmd, timeout=15):
        b_cmd = cmd.encode(encoding=self.coding)
        try:
            b_outs, errs = self.popen.communicate(b_cmd, timeout=timeout)
        except sp.TimeoutExpired:
            self.popen.kill()
            b_outs, errs = self.popen.communicate()
        outs = b_outs.decode(encoding=self.coding)
        return outs, errs

    @staticmethod
    def _where(filename, dirs=None, env="PATH"):
        """Find file in current dir, in deep_lookup cache or in system path"""
        if dirs is None:
            dirs = []
        if not isinstance(dirs, list):
            dirs = [dirs]
        if glob(filename):
            return filename
        paths = [os.curdir] + os.environ[env].split(os.path.pathsep) + dirs
        try:
            return next(os.path.normpath(match)
                        for path in paths
                        for match in glob(os.path.join(path, filename))
                        if match)
        except (StopIteration, RuntimeError):
            raise IOError("File not found: %s" % filename)
class Fly:
    def __init__(self):
        self.tanks_location = [[8.91304348,1.20652174,0.61669004],
                               [6.91304348,-1.39347826,0.21669004],
                               [-1.68695652,1.20652174,-0.28330996],
```

```
                                [3.11304348,0.60652174,-0.18330996],
                                [-5.28695652,-0.29347826,0.41669004],
                                [-2.08695652,-1.49347826,0.21669004]]
        self.tanks_size     = [[1.5, 0.9, 0.3],
                                [2.2, 0.8, 1.1],
                                [2.4,1.1,0.9],
                                [1.7,1.3,1.2],
                                [2.4,1.2,1],
                                [2.4,1,0.5]]
        self.tanks_rest     = [0.3,1.5,2.1,1.9,2.6,0.8]
        self.speed = [1.1, 1.8, 1.7, 1.5, 1.6, 1.1]
        self.mess = 3000
        self.density = 850

    def calEcomsume(self, comsume):
        ecomsume = [0] * 6
        for i in range(6):
            ecomsume[i] = comsume[i]
        ecomsume[1] -= comsume[0]
        ecomsume[4] -= comsume[5]
        return ecomsume

    def UpdateComsume(self,ecomsume,time):
        for i in range(len(ecomsume)):
            self.tanks_rest[i] -= ecomsume[i]*time / self.density
            if self.tanks_rest[i] < 0 :
                if abs(self.tanks_rest[i]) > 1e-5:
                    print("something wrong")
                else:
                    self.tanks_rest[i] = 0

    def calC(self):
        local = [0,0,0]
        for i in range(6):
            for j in range(2):
                local[j] += self.tanks_rest[i] * self.density * self.tanks_location[i][
    j]
        for i in range(6):
            j = 2
            local[j] += self.tanks_rest[i] * self.density * (self.tanks_location[i][j]
    - self.tanks_size[i][j]/2 +
                                                    self.tanks_rest[i] / (self.tanks_s
    ize[i][0]*self.tanks_size[i][1]*2))
        all_mess = self.mess
```

```
100.         for i in range(6):
101.             all_mess += self.tanks_rest[i] * self.density
102.         return [x / all_mess for x in local]
103.
104. def writefile(location_t, last, demand, pnums=None):
105.     reference = open('reference.lg4','r')
106.     file_t = open('2.lg4', 'w')
107.     ref_lines = reference.readlines()
108.     for ref_line in ref_lines:
109.         if len(ref_line.split()) == 0:
110.             pass
111.         elif ref_line.split()[0] ==  'LOCATION':
112.             # print(location_t[0])
113.             print('LOCATION = %s, %s, %s;' %(format(location_t[0],'.10e'),format(locat
    ion_t[1],'.10e'),format(location_t[2],'.10e')), file=file_t)
114.         elif ref_line.split()[0] == 'LAST':
115.             print('LAST = %s, %s, %s, %s, %s, %s;' %(format(last[0],'.10e'),format(las
    t[1],'.10e'),format(last[2],'.10e'),format(last[3],'.10e'),format(last[4],'.10e'),forma
    t(last[5],'.10e')), file=file_t)
116.         elif ref_line.split()[0] == 'DEMAND':
117.             print('DEMAND = %s;' %(format(demand,'.10e')), file=file_t)
118.         elif ref_line.split()[0] == 'enddata':
119.             # for i in range(len(pnums)):
120.             #     if pnums[i] != 0 and pnums[i] < 60:
121.             #         print('PICKUP(%d) = 1;' %(i+1), file=file_t)
122.             if pnums:
123.                 print('PICKUP ='+' '.join([str(pnum) for pnum in pnums])+';',file=file
    _t)
124.             print(ref_line,file=file_t)
125.         else:
126.             print(ref_line,file=file_t)
127.     file_t.close()
128.     reference.close()
129.
130. def readfile():
131.     file = open('result_2.txt','r')
132.     filelines = file.readlines()
133.     comsume = []
134.     for fileline in filelines:
135.         if fileline.find('COMSUME') != -1:
136.             if fileline.find('ECOMSUME') == -1:
137.                 comsume.append(float(fileline.split()[2]))
138.     pickup = []
139.     for fileline in filelines:
```

```
        if fileline.find('PICKUP') != -1:
            pickup.append(int(float(fileline.split()[2])))
    return comsume, pickup

def readexcel():
    path = 'C:/Users/PaUlGuO/Desktop/model/team'
    wb = openpyxl.load_workbook(path + '/附件 3-问题 2 数据.xlsx')
    ws1 = wb[wb.sheetnames[0]]
    ws2 = wb[wb.sheetnames[1]]
    all_comsume = []
    all_location = []
    for v in ws1['B2':'B7201']:
        all_comsume.append(v[0].value)
    for i in range(2,7202):
        tmp = []
        for j in range(2,5):
            tmp.append(ws2.cell(row=i, column=j).value)
        all_location.append(tmp)
    return all_comsume, all_location

def process_bar(percent, start_str='', end_str='', total_length=0):
    bar = ''.join(["\033[31m%s\033[0m"%'   '] * int(percent * total_length)) + ''
    bar = '\r' + start_str + bar.ljust(total_length) + ' {:0>4.1f}%|'.format(percent*100) + end_str
    print(bar, end='', flush=True)

def opt():

    fly = Fly()

    all_comsume, all_location = readexcel()

    results = openpyxl.Workbook()
    result = results.active

    for i in tqdm(range(len(all_location))):
        # v1.0
        # if i<=90:
        #     pnums = None
        # elif i <= 750:
        #     pnums = [1,1,1,0,0,0]
        # elif i <= 1000:
        #     pnums = [1,1,0,0,1,0]
        # elif i <= 1500:
```

```
183.         #     pnums = [1,1,1,0,0,0]
184.         # elif i <= 1904:
185.         #     pnums = None
186.         # elif i <= 3400:
187.         #     pnums = [0,0,1,1,0,0]
188.         # # elif i <= 4000:
189.         # #     pnums = [0,0,1,0,1,0]
190.         # else:
191.         #     pnums = None
192.
193.         # v2.0
194.         if i <= 90:
195.             pnums = None
196.         elif i <= 380:
197.             pnums = [0,1,0,0,0,0]
198.         elif i <= 870:
199.             pnums = [1,1,0,0,0,0]
200.         elif i <= 1420:
201.             pnums = [0,1,0,0,0,0]
202.         elif i <= 2100:
203.             pnums = [0,1,0,1,0,0]
204.         elif i <= 2800:
205.             pnums = [0,0,0,1,0,0]
206.         elif i <= 3340:
207.             pnums = [0,0,1,1,0,0]
208.         elif i <= 3950:
209.             pnums = [0,0,1,0,0,0]
210.         elif i <= 5250:
211.             pnums = [0,0,1,0,1,1]
212.         else :
213.             pnums = [0,0,0,0,1,1]
214.
215.         if i > 6000:
216.             continue
217.
218.         writefile(all_location[i], [rest * fly.density for rest in fly.tanks_rest], al
    l_comsume[i], pnums)
219.
220.         with PowerShell('GBK') as ps:
221.             outs, errs = ps.run('runlingo 2.ltf')
222.
223.         comsume, pickup = readfile()
224.
225.         for j in range(len(comsume)):
```

```
            comsume[j] *= pickup[j]

        fly.UpdateComsume(fly.calEcomsume(comsume),1)

        result.append(comsume + all_location[i] + fly.calC())

        if i % 1000 == 0:
            results.save('result2_%d.xlsx' %(i))

    results.save('result2.xlsx')

if __name__ == '__main__':
    # Example:

    opt()

    # print()
```

[求解逐时间节点的供油最优解的 LINGO 代码(样例)]

```
MODEL:
    sets:
        locations/1..3/:LOCATION;
        comsumer/1..6/:COMSUME,PICKUP,SPEED,ECOMSUME,LAST,SUM,SUM_C;
        shape/1..3/:SIZE;
        links_1(comsumer,locations):BOXS_L,BOXS_C;
        links_2(comsumer,shape):BOXS_S;
        delta/1..3/:DL;
    endsets

    MIN = @SUM(delta(I):DL(I)^2);
    DL(1) = LOCATION(1) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*BOXS_L(I,1)/(@SUM(comsumer(J):LAST(J)-ECOMSUME(J)) + MESS));
    DL(2) = LOCATION(2) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*BOXS_L(I,2)/(@SUM(comsumer(J):LAST(J)-ECOMSUME(J)) + MESS));
    DL(3) = LOCATION(3) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*(BOXS_L(I,3) - BOXS_S(I,3)/2 + (LAST(I)-ECOMSUME(I))/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/(@SUM(comsumer(I):LAST(I)-ECOMSUME(I))+MESS);

    @FOR(comsumer(I):LAST(I) >= ECOMSUME(I));
    !DL(1) = LOCATION(1) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*BOXS_L(I,1)/(@SUM(comsumer(J):LAST(J)) + MESS));
    !DL(2) = LOCATION(2) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*BOXS_L(I,2)/(@SUM(comsumer(J):LAST(J)) + MESS));
```

```
    !DL(3) = LOCATION(3) - @SUM(comsumer(I):(LAST(I)-ECOMSUME(I))*(BOXS_L(I,3) - BOXS_S(I,3)/2 + (LAST(I)-ECOMSUME(I))/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/(@SUM(comsumer(I):LAST(I))+MESS);

    @FOR(comsumer(I)|I#NE#2 #AND# I#NE#5:
        ECOMSUME(I) = COMSUME(I)*PICKUP(I));
    ECOMSUME(2) = COMSUME(2)*PICKUP(2) - COMSUME(1)*PICKUP(1);
    ECOMSUME(5) = COMSUME(5)*PICKUP(5) - COMSUME(6)*PICKUP(6);
    @SUM(comsumer(I):PICKUP(I)) <= 3;
    @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:PICKUP(I)) <= 2;
    @FOR(comsumer(I):COMSUME(I)<=SPEED(I));
    !@FOR(comsumer(I):COMSUME(I) = COMSUME(I)*PICKUP(I));
    @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:COMSUME(I) * PICKUP(I)) >= DEMAND;
    @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:COMSUME(I) * PICKUP(I)) <= 1.2 * DEMAND;
    @FOR(comsumer(I):@BIN(PICKUP(I)));
    @FOR(comsumer(I):@free(ECOMSUME(I)));
    @FOR(delta(I):@free(DL(I)));

    data:
        LOCATION = -0.000006423592, 0.0000012957, -0.000000478744;
        SPEED = 1.1, 1.8, 1.7, 1.5, 1.6, 1.1;
        DENSITY = 850;
        BOXS_L =    8.91304348  1.20652174  0.61669004
                    6.91304348 -1.39347826  0.21669004
                   -1.68695652  1.20652174 -0.28330996
                    3.11304348  0.60652174 -0.18330996
                   -5.28695652 -0.29347826  0.41669004
                   -2.08695652 -1.49347826  0.21669004;
        BOXS_S =    1.5 0.9 0.3
                2.2 0.8 1.1
                2.4 1.1 0.9
                1.7 1.3 1.2
                2.4 1.2 1
                2.4 1   0.5;
        LAST = 255, 1275, 1785, 1615, 2210, 680;
        MESS = 3000;
        DEMAND = 0.0100558899754559;
    enddata
end
```

问题 3：

[剩余油量求解的 LINGO 代码(样例)]

```
MODEL:

```

```
    sets:

        locations/1..3/:LOCATION2,LOCATION0,LOCATION1;

        comsumer/1..6/:CAPACITY,VOLUME,VOLUME_C1,VOLUME_C2,CHARGE1,CHARGE2;

        shape/1..3/:SIZE;

        links_1(comsumer,locations):BOXS_L,BOXS_C;

        links_2(comsumer,shape):BOXS_S;

        delta/1..3/:DL2,DL0,DL1;

    endsets

    !MIN = @SUM(delta(I):DL2(I)^2 + DL0(I)^2 + DL1(I)^2);
    MIN = @SMAX(@SUM(delta(I):DL1(I)^2),@SUM(delta(I):DL0(I)^2),@SUM(delta(I):DL2(I)^2)
);

    DL2(1) = LOCATION2(1) - @SUM(comsumer(I):VOLUME_C2(I)*BOXS_L(I,1))/(MESS + @SUM(com
sumer(I)|I#GT#1 #AND# I#LT#6:CHARGE2(I)));
    DL2(2) = LOCATION2(2) - @SUM(comsumer(I):VOLUME_C2(I)*BOXS_L(I,2))/(MESS + @SUM(com
sumer(I)|I#GT#1 #AND# I#LT#6:CHARGE2(I)));
    DL2(3) = LOCATION2(3) - @SUM(comsumer(I):VOLUME_C2(I)*(BOXS_L(I,3) - BOXS_S(I,3)/2
+ VOLUME_C2(I)/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/(MESS + @SUM(comsumer(I)|I#GT#1 #A
ND# I#LT#6:CHARGE2(I)));

    DL0(1) = LOCATION0(1) - @SUM(comsumer(I):VOLUME(I)*BOXS_L(I,1))/MESS;
    DL0(2) = LOCATION0(2) - @SUM(comsumer(I):VOLUME(I)*BOXS_L(I,2))/MESS;
    DL0(3) = LOCATION0(3) - @SUM(comsumer(I):VOLUME(I)*(BOXS_L(I,3) - BOXS_S(I,3)/2 + V
OLUME(I)/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/MESS;

    DL1(1) = LOCATION1(1) - @SUM(comsumer(I):VOLUME_C1(I)*BOXS_L(I,1))/(MESS + @SUM(com
sumer(I)|I#GT#1 #AND# I#LT#6:CHARGE1(I)));
    DL1(2) = LOCATION1(2) - @SUM(comsumer(I):VOLUME_C1(I)*BOXS_L(I,2))/(MESS + @SUM(com
sumer(I)|I#GT#1 #AND# I#LT#6:CHARGE1(I)));
    DL1(3) = LOCATION1(3) - @SUM(comsumer(I):VOLUME_C1(I)*(BOXS_L(I,3) - BOXS_S(I,3)/2
+ VOLUME_C1(I)/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/(MESS + @SUM(comsumer(I)|I#GT#1 #A
ND# I#LT#6:CHARGE1(I)));

    @FOR(comsumer(I):VOLUME_C1(I) <= CAPACITY(I));
    @FOR(comsumer(I):VOLUME_C2(I) <= CAPACITY(I));
    @FOR(comsumer(I):CAPACITY(I) = DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*BOXS_S(I,3));
```

```
    @SUM(comsumer(I):VOLUME(I)) = 850 ;
    @FOR(comsumer(I):VOLUME_C1(I) = VOLUME(I) + CHARGE1(I));
    @FOR(comsumer(I):VOLUME_C2(I) = VOLUME(I) + CHARGE2(I));

    @FOR(delta(I):@free(DL1(I)));
    @FOR(delta(I):@free(DL0(I)));
    @FOR(delta(I):@free(DL2(I)));

    data:

        LOCATION0  = -0.269845557421, -0.134439076516, -0.08153096928;
        LOCATION1 = -0.269842576744, -0.134441897023, -0.081529893624;
        LOCATION2 = -0.284694621466, -0.120387944621, -0.120387944621;

        DENSITY = 850;

        BOXS_L =    8.91304348  1.20652174  0.61669004

                    6.91304348 -1.39347826  0.21669004

                   -1.68695652  1.20652174 -0.28330996

                    3.11304348  0.60652174 -0.18330996

                   -5.28695652 -0.29347826  0.41669004

                   -2.08695652 -1.49347826  0.21669004;

        BOXS_S =    1.5 0.9 0.3

                2.2 0.8 1.1

                2.4 1.1 0.9

                1.7 1.3 1.2

                2.4 1.2 1

                2.4 1   0.5;

        MESS = 3850;
        CHARGE1 = 0,0,0,0,0.01,0;
        CHARGE2 = 0,0,59.1289426,0,1.29753213,15.7803228;
    enddata
```

```
81.
82. end
```

[反向求解逐时间节点供油最优解的 LINGO 代码]

```
1.  MODEL:
2.      sets:
3.          locations/1..3/:LOCATION;
4.          comsumer/1..6/:COMSUME,PICKUP,SPEED,ECOMSUME,LAST,SUM,SUM_C;
5.          shape/1..3/:SIZE;
6.          links_1(comsumer,locations):BOXS_L,BOXS_C;
7.          links_2(comsumer,shape):BOXS_S;
8.          delta/1..3/:DL;
9.      endsets
10.
11.     MIN = @SUM(delta(I):DL(I)^2);
12.     DL(1) = LOCATION(1) - @SUM(comsumer(I):(LAST(I)+ECOMSUME(I))*BOXS_L(I,1)/(@SUM(coms
    umer(J):LAST(J)+ECOMSUME(J)) + MESS));
13.     DL(2) = LOCATION(2) - @SUM(comsumer(I):(LAST(I)+ECOMSUME(I))*BOXS_L(I,2)/(@SUM(coms
    umer(J):LAST(J)+ECOMSUME(J)) + MESS));
14.     DL(3) = LOCATION(3) - @SUM(comsumer(I):(LAST(I)+ECOMSUME(I))*(BOXS_L(I,3) - BOXS_S(
    I,3)/2 + (LAST(I)+ECOMSUME(I))/(DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*2)))/(@SUM(comsumer(I):
    LAST(I)+ECOMSUME(I))+MESS);
15.
16.     @FOR(comsumer(I)|I#NE#2 #AND# I#NE#5:
17.         ECOMSUME(I) = COMSUME(I)*PICKUP(I));
18.     ECOMSUME(2) = COMSUME(2)*PICKUP(2) - COMSUME(1)*PICKUP(1);
19.     ECOMSUME(5) = COMSUME(5)*PICKUP(5) - COMSUME(6)*PICKUP(6);
20.     @SUM(comsumer(I):PICKUP(I)) <= 3;
21.     @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:PICKUP(I)) <= 2;
22.     @FOR(comsumer(I):COMSUME(I)<=SPEED(I));
23.     @FOR(comsumer(I):COMSUME(I) = COMSUME(I)*PICKUP(I));
24.     @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:COMSUME(I) * PICKUP(I)) >= DEMAND;
25.     @SUM(comsumer(I)|I#GT#1 #AND# I#LT#6:COMSUME(I) * PICKUP(I)) <= DEMAND;
26.     @FOR(comsumer(I):@BIN(PICKUP(I)));
27.     @FOR(comsumer(I):@free(ECOMSUME(I)));
28.     @FOR(delta(I):@free(DL(I)));
29.     @FOR(comsumer(I):LAST(I) + ECOMSUME(I) >=0);
30.     @FOR(comsumer(I):LAST(I) + ECOMSUME(I) <= DENSITY*BOXS_S(I,1)*BOXS_S(I,2)*BOXS_S(I,
    3));
31.     data:
32.         LOCATION = -0.000006423592, 0.0000012957, -0.000000478744;
33.         SPEED = 1.1, 1.8, 1.7, 1.5, 1.6, 1.1;
34.         DENSITY = 850;
```

```
        BOXS_L =    8.91304348  1.20652174  0.61669004
                    6.91304348 -1.39347826  0.21669004
                   -1.68695652  1.20652174 -0.28330996
                    3.11304348  0.60652174 -0.18330996
                   -5.28695652 -0.29347826  0.41669004
                   -2.08695652 -1.49347826  0.21669004;
        BOXS_S =    1.5 0.9 0.3
                2.2 0.8 1.1
                2.4 1.1 0.9
                1.7 1.3 1.2
                2.4 1.2 1
                2.4 1   0.5;
        LAST = 255, 1275, 1785, 1615, 2210, 680;
        MESS = 3000;
        DEMAND = 0.0100558899754559;
    enddata
end
```

[反向求解逐时间节点供油最优解的 python 控制代码]

```python
## for dt = 1 without constrain of 60s

import os
from glob import glob
import subprocess as sp
import sys
import openpyxl
import time
from tqdm import tqdm
class PowerShell:
    # from scapy
    def __init__(self, coding, ):
        cmd = [self._where('PowerShell.exe'),
               "-NoLogo", "-NonInteractive",  # Do not print headers
               "-Command", "-"]  # Listen commands from stdin
        startupinfo = sp.STARTUPINFO()
        startupinfo.dwFlags |= sp.STARTF_USESHOWWINDOW
        self.popen = sp.Popen(cmd, stdout=sp.PIPE, stdin=sp.PIPE, stderr=sp.STDOUT, sta
rtupinfo=startupinfo)
        self.coding = coding

    def __enter__(self):
        return self

    def __exit__(self, a, b, c):
```

```
        self.popen.kill()

    def run(self, cmd, timeout=15):
        b_cmd = cmd.encode(encoding=self.coding)
        try:
            b_outs, errs = self.popen.communicate(b_cmd, timeout=timeout)
        except sp.TimeoutExpired:
            self.popen.kill()
            b_outs, errs = self.popen.communicate()
        outs = b_outs.decode(encoding=self.coding)
        return outs, errs

    @staticmethod
    def _where(filename, dirs=None, env="PATH"):
        """Find file in current dir, in deep_lookup cache or in system path"""
        if dirs is None:
            dirs = []
        if not isinstance(dirs, list):
            dirs = [dirs]
        if glob(filename):
            return filename
        paths = [os.curdir] + os.environ[env].split(os.path.pathsep) + dirs
        try:
            return next(os.path.normpath(match)
                        for path in paths
                        for match in glob(os.path.join(path, filename))
                        if match)
        except (StopIteration, RuntimeError):
            raise IOError("File not found: %s" % filename)
class Fly:
    def __init__(self):
        self.tanks_location = [[8.91304348,1.20652174,0.61669004],
                               [6.91304348,-1.39347826,0.21669004],
                               [-1.68695652,1.20652174,-0.28330996],
                               [3.11304348,0.60652174,-0.18330996],
                               [-5.28695652,-0.29347826,0.41669004],
                               [-2.08695652,-1.49347826,0.21669004]]
        self.tanks_size     = [[1.5, 0.9, 0.3],
                               [2.2, 0.8, 1.1],
                               [2.4,1.1,0.9],
                               [1.7,1.3,1.2],
                               [2.4,1.2,1],
                               [2.4,1,0.5]]
        self.tanks_rest     = [0.3,1.5,2.1,1.9,2.6,0.8]
```

```
69.         self.speed = [1.1, 1.8, 1.7, 1.5, 1.6, 1.1]
70.         self.mess = 3000
71.         self.density = 850
72.         self.volume = []
73.         for i in range(6):
74.             v = 1
75.             for j in range(3):
76.                 v *= self.tanks_size[i][j]
77.             self.volume.append(v)
78.
79.     def calEcomsume(self, comsume):
80.         ecomsume = [0] * 6
81.         for i in range(6):
82.             ecomsume[i] = comsume[i]
83.         ecomsume[1] -= comsume[0]
84.         ecomsume[4] -= comsume[5]
85.         return ecomsume
86.
87.     def UpdateComsume(self,ecomsume,time):
88.         rets = [False] * 6
89.         for i in range(len(ecomsume)):
90.             self.tanks_rest[i] -= ecomsume[i]*time / self.density
91.             if self.tanks_rest[i] < 0 :
92.                 if abs(self.tanks_rest[i]) > 1e-5:
93.                     # print("something wrong")
94.                     self.tanks_rest[i] = 0
95.                     rets[i] = True
96.                 else:
97.                     self.tanks_rest[i] = 0
98.         return rets
99.
100.     def UpdataCharge(self,ecomsume,time):
101.         rets = [False] * 6
102.         for i in range(len(ecomsume)):
103.             self.tanks_rest[i] += ecomsume[i] * time / self.density
104.             if self.tanks_rest[i] > self.volume[i]:
105.                 rets[i] = True
106.         return rets
107.
108.     def UpdateRest(self,init_r):
109.         for i in range(len(init_r)):
110.             self.tanks_rest[i] = init_r[i] / self.density
111.             # self.tanks_rest[i] = init_r[i]
112.
```

```
113.     def calC(self):
114.         local = [0,0,0]
115.         for i in range(6):
116.             for j in range(2):
117.                 local[j] += self.tanks_rest[i] * self.density * self.tanks_location[i]
    [j]
118.         for i in range(6):
119.             j = 2
120.             local[j] += self.tanks_rest[i] * self.density * (self.tanks_location[i][j]
     - self.tanks_size[i][j]/2 +
121.                                                              self.tanks_rest[i] / (self.tanks_
    size[i][0]*self.tanks_size[i][1]*2))
122.         all_mess = self.mess
123.         for i in range(6):
124.             all_mess += self.tanks_rest[i] * self.density
125.         return [x / all_mess for x in local]
126.
127. def writefile(location_t, last, demand, pnums=None):
128.     reference = open('reference3.lg4','r')
129.     file_t = open('3.lg4', 'w')
130.     ref_lines = reference.readlines()
131.     for ref_line in ref_lines:
132.         if len(ref_line.split()) == 0:
133.             pass
134.         elif ref_line.split()[0] ==  'LOCATION':
135.             # print(location_t[0])
136.             print('LOCATION = %s, %s, %s;' %(format(location_t[0],'.10e'),format(locat
    ion_t[1],'.10e'),format(location_t[2],'.10e')), file=file_t)
137.         elif ref_line.split()[0] == 'LAST':
138.             print('LAST = %s, %s, %s, %s, %s, %s;' %(format(last[0],'.10e'),format(las
    t[1],'.10e'),format(last[2],'.10e'),format(last[3],'.10e'),format(last[4],'.10e'),forma
    t(last[5],'.10e')), file=file_t)
139.         elif ref_line.split()[0] == 'DEMAND':
140.             print('DEMAND = %s;' %(format(demand,'.10e')), file=file_t)
141.         elif ref_line.split()[0] == 'enddata':
142.             # for i in range(len(pnums)):
143.             #     if pnums[i] != 0 and pnums[i] < 60:
144.             #         print('PICKUP(%d) = 1;' %(i+1), file=file_t)
145.             if pnums:
146.                 print('PICKUP ='+' '.join([str(pnum) for pnum in pnums])+';',file=file
    _t)
147.             print(ref_line,file=file_t)
148.         # elif ref_line.split()[0] == '@FOR(delta(I):@free(DL(I)));':
149.         #     print(ref_line,file=file_t)
```

```
150.         #     for i in range(len(present)):
151.         #         if present[i] == init_l[i]:
152.         #             print('LAST(%d) >= ECOMSUME(%d);'%(i+1,i+1),file = file_t)
153.         else:
154.             print(ref_line,file=file_t)
155.     file_t.close()
156.     reference.close()
157. 
158. def writeref(present):
159.     reference = open('initial_3_ref.lg4','r')
160.     file_t = open('initial_3.lg4', 'w')
161.     ref_lines = reference.readlines()
162.     for ref_line in ref_lines:
163.         if len(ref_line.split()) == 0:
164.             pass
165.         elif ref_line.split()[0] == '@FOR(comsumer(I):VOLUME(I)':
166.             print(ref_line,file=file_t)
167.             for i in range(1,7):
168.                 print('VOLUME(%d) >= %f * CAPACITY(%d);'%(i,present[i-1],i),file=file_
    t)
169.         else:
170.             print(ref_line,file=file_t)
171.     reference.close()
172.     file_t.close()
173. 
174. def readfile():
175.     file = open('result_3.txt','r')
176.     filelines = file.readlines()
177.     comsume = []
178.     for fileline in filelines:
179.         if fileline.find('COMSUME') != -1:
180.             if fileline.find('ECOMSUME') == -1:
181.                 comsume.append(float(fileline.split()[2]))
182.     pickup = []
183.     for fileline in filelines:
184.         if fileline.find('PICKUP') != -1:
185.             pickup.append(int(float(fileline.split()[2])))
186.     return comsume, pickup
187. 
188. def readref():
189.     file = open('initial_3.txt','r')
190.     filelines = file.readlines()
191.     volume = []
192.     for fileline in filelines:
```

```
193.         if fileline.find('VOLUME') != -1:
194.             volume.append(float(fileline.split()[2]))
195.     return volume
196. 
197. def readexcel():
198.     path = 'C:/Users/PaUlGuO/Desktop/model/team'
199.     wb = openpyxl.load_workbook(path + '/附件 4-问题 3 数据.xlsx')
200.     ws1 = wb[wb.sheetnames[0]]
201.     ws2 = wb[wb.sheetnames[1]]
202.     all_comsume = []
203.     all_location = []
204.     for v in ws1['B2':'B7201']:
205.         all_comsume.append(v[0].value)
206.     for i in range(2,7202):
207.         tmp = []
208.         for j in range(2,5):
209.             tmp.append(ws2.cell(row=i, column=j).value)
210.         all_location.append(tmp)
211.     return all_comsume, all_location
212. 
213. def process_bar(percent, start_str='', end_str='', total_length=0):
214.     bar = ''.join(["\033[31m%s\033[0m"%'   '] * int(percent * total_length)) + ''
215.     bar = '\r' + start_str + bar.ljust(total_length) + ' {:0>4.1f}%|'.format(percent*1
    00) + end_str
216.     print(bar, end='', flush=True)
217. 
218. def opt():
219. 
220.     fly = Fly()
221. 
222.     all_comsume, all_location = readexcel()
223. 
224.     all_comsume.reverse()
225.     all_location.reverse()
226. 
227.     # holdon = [False] * 6
228.     init_l = [0.0] * 6
229.     # init_r = [1.0] * 6
230. 
231.     for iter in range(1):
232. 
233.         # present = [(init_r[i] + init_l[i]) / 2 if holdon[i] else 0.0 for i in range(
    len(init_l))]
234. 
```

```
235.         # for i in range(1,len(present)-1):
236.         #     if present[i] >= 0.97:
237.         #         present[i] = 1.0
238.         #         init_l[i] = 1.0
239.         #         init_r[i] = 1.0
240.
241.         print('********iter %d*********' %(iter+1))
242.
243.         # print(init_l)
244.         # print(init_r)
245.         # print(present)
246.
247.         # present_r = solve_init(holdon, init_r)
248.         # f = open('init_3.txt','a')
249.         # print('********iter %d*********' %(i+1),file = f)
250.         # print(' '.join([str(r) for r in present_r]), file = f)
251.         # f.close()
252.
253.         results = openpyxl.Workbook()
254.         result = results.active
255.
256.         # init_r = [0,54,137,111,120,428]
257.         # init_r = [0,0.063975118,0.188441394,0.133738096,0.1428346,0.503529412]
258.         # init_r = [287.0330, 1159.560, 1077.014, 2253.213, 2194.129, 684.0513]
259.         # init_r = [344.2500, 1640.684, 1911.173, 1156.975, 2066.856, 535.0624]
260.         # writeref(init_l) ##
261.         # with PowerShell('GBK') as ps:
262.         #     outs, errs = ps.run('runlingo 3_ref.ltf')
263.         # init_rest = readref()
264.         # init_rest = [0.0,48.09765,147.5565,118.8232,115.1304,420.3923]
265.         # init_rest = [0.0,48.62045,149.6112,119.3313,109.5266,422.9104]
266.         init_rest = [0.0,47.18759,131.5116,144.8741,130.9014,395.5253]
267.         fly.UpdateRest(init_rest)
268.
269.         maxx = 0
270.         # print(present)
271.         for i in tqdm(range(len(all_location)-1)):
272.
273.             if all_comsume[i] == 0:
274.                 result.append([i+1])
275.                 continue
276.
277.             else:
278.                 pnums = None
```

```
279.             # print('halo')
280.             # pnums = None
281.             # writefile(all_location[94], [rest * fly.density for rest in fly.tanks_re
    st], all_comsume[94], [0]*6 )
282.             writefile(all_location[i+1], [rest * fly.density for rest in fly.tanks_res
    t], all_comsume[i], pnums)
283.
284.             # print(i + 1)
285.
286.             with PowerShell('GBK') as ps:
287.                 outs, errs = ps.run('runlingo 3.ltf')
288.
289.             comsume, pickup = readfile()
290.
291.             if len(comsume) == 0:
292.                 break
293.
294.             for j in range(len(comsume)):
295.                 comsume[j] *= pickup[j]
296.
297.             rets = fly.UpdataCharge(fly.calEcomsume(comsume),1)
298.
299.             result.append([i+1] + comsume + all_location[i+1] + fly.calC() + fly.tanks
    _rest)
300.
301.             C1 = fly.calC()
302.             C0 = all_location[i+1]
303.             tmp = 0
304.
305.             for k in range(len(C1)):
306.                 tmp += (C1[k] - C0[k]) ** 2
307.
308.             maxx = max(tmp,maxx)
309.             # if tmp > 0.05 : ## 质心偏移上限
310.             #     break
311.
312.             # if True in rets:
313.
314.             #     break
315.
316.         results.save('result3_%d_last.xlsx' %(iter+1))
317.         # log = open('log.txt','a')
318.         # print('********* iter %d *********' %(iter+1), file = log)
319.         # print(rets, file= log)
```

```
        # print('present constraint', file = log)
        # print(present, file = log)
        # log.close()
        # if True not in rets: ## 二分思路
        #     for i in range(len(present)):
        #         init_r[i] = (init_r[i] + present[i]) / 2.0
        # else:
        #     for i in range(len(rets)):
        #         if rets[i]:
        #             if init_l[i] == 1.0 :
        #                 print('there is something wrong')
        #                 break
        #             init_l[i] = present[i]
        #             used[i] = True
        #         else:
        #             if used[i]:
        #                 init_l[i] = (present[i] + init_l[i]) - init_r[i]
        #                 used[i] = False
        # for i in range(len(init_r)):
        #     if abs(init_r[i] - init_l[i]) < 0.001:
        #         init_r[i] = init_l[i]
    print(i)

    print(maxx)

if __name__ == '__main__':
    # Example:

    opt()

    # print()
```

问题 4：
[基于 60s 时间段的供油最优策略 matlab 代码]

```
clc
clear
warning off
[consume]=xlsread('C:\Users\PaUlGuO\Desktop\model\team\附件 5-问题 4 数据.xlsx',1,'B2:B7201');
[angle]=xlsread('C:\Users\PaUlGuO\Desktop\model\team\附件 5-问题 4 数据.xlsx',2,'B2:B7201');
%final = ones(7200,1);
fv = zeros(7200,6);
At = [1e20,1e20,0,1e20,1e20,1e20;
```

```
    1e20,1e20,1e20,0,1e20,1e20;
    1e20,1e20,1e20,1e20,0,1e20;
    1e20,0,1e20,1e20,1e20,1e20;
    0,0,1e20,1e20,1e20,1e20;
    1e20,0,1e20,1e20,1e20,0;
    0,0,1e20,1e20,1e20,0;
    0,1e20,0,1e20,1e20,1e20;
    0,1e20,0,1e20,1e20,0;
    1e20,1e20,0,1e20,1e20,0;
    1e20,1e20,1e20,0,1e20,0;
    0,1e20,1e20,0,1e20,0;
    0,1e20,1e20,0,1e20,1e20;
    1e20,1e20,1e20,1e20,0,0;
    0,1e20,1e20,1e20,0,1e20;
    0,1e20,1e20,1e20,0,0;
    1e20,0,0,1e20,1e20,1e20;
    1e20,0,1e20,0,1e20,1e20;
    1e20,0,1e20,1e20,0,1e20;
    1e20,1e20,0,0,1e20,1e20;
    1e20,1e20,0,1e20,0,1e20;
    1e20,1e20,1e20,0,0,1e20;
    0,0,0,1e20,1e20,1e20;
    0,0,1e20,0,1e20,1e20;
    0,0,1e20,1e20,0,1e20;
    0,1e20,0,0,1e20,1e20;
    0,1e20,0,1e20,0,1e20;
    0,1e20,1e20,0,0,1e20;
    1e20,0,0,1e20,1e20,0;
    1e20,0,1e20,0,1e20,0;
    1e20,0,1e20,1e20,0,0;
    1e20,1e20,0,0,1e20,0;
    1e20,1e20,0,1e20,0,0;
    1e20,1e20,1e20,0,0,0;
    ];
%last = [255, 1275, 1785, 1615, 2210, 680];
%last = [255,1275,1.522511111320173e+03,1.428100874944118e+03,2.281296163866746e+03,3.7
53451648073769e+02];
%last = [2.206478555793609e+02,1.036794773504075e+03,1.214321966570699e+03,19.219643564
536412,1.353997304699008e+03,1.914292078186719e-07];
%last = [2.206478555793609e+02,5.278026909538892e+02,6.217837833712356e+02,19.219643564
536412,9.947898160819083e+02,1.914292078186719e-07];
%last = [2.206478555793609e+02,2.356128425237355e+02,2.018039531832358e+02,19.219643564
536412,6.789007305804507e+02,1.914292078186719e-07];
```

```
last = [1.768987624794272e+02,1.350202177198442e+02,40.453142352864720,19.219643564536412,4.976543986285903e+02,1.914292078186719e-07];
Beq = 0;
v0 = [0,0,0,0,0,0];
LB = [0,0,0,0,0,0];
UB0 = [1.1,1.8,1.7,1.5,1.6,1.1];
UB = [1.1,1.8,1.7,1.5,1.6,1.1];
for n=1:6
    UB(n) = min(last(n),UB0(n));
end
A = [0,-1,-1,-1,-1,0;0,1,1,1,1,0];
options = optimset('TolX',1e-8,'TolFun',1e-8, 'MaxIter', 100, 'MaxFunEvals',1000,'diffminchange', 1e-10);
for j=0:60:6000
    dissum = zeros(34:1);%用来记录每种方案的距离和
    vtmp = zeros(34,60,6);%用来记录每种方案每秒 6 个油箱的供油速度
    for i=1:1:34
        lasttmp = last;
        Aeq = At(i,:);
        distmp = zeros(60,1);%用来记录每种情况 60 秒的距离
        for k=1:60
            B = [-1*consume(k+j-1);consume(k+j-1)*1];
            [vt, dis] = fmincon(@(v)ques4sol(v,lasttmp,angle(k+j-1)),v0,A,B,Aeq,Beq,LB,UB,'ques2cons',options);
%             if sum(vt)<consume(k+j-1)
%                 dissum(i) = 100;
%                 break;
%             end
            for m=1:6
                if vt(m)<1e-7
                    vt(m) = 0;
                end
            end
            vtmp(i,k,:)=vt;
            distmp(k,1)=dis;
            lasttmp = lasttmp - vt;
            lasttmp(2) = lasttmp(2) + vt(1);
            lasttmp(5) = lasttmp(5) + vt(6);
            for n=1:6
                UB(n) = min(lasttmp(n),UB0(n));
            end
        end
        dissum(i,1) = sum(sum(distmp));
    end
```

```
    [p,q] = min(dissum);
    %final(j:j+59) = ;
    fv(j:j+59,:) = vtmp(q,:,:);
    last = last - sum(fv(j:j+59,:));
    last(2) = last(2) + sum(fv(j:j+59,1));
    last(5) = last(5) + sum(fv(j:j+59,6));
    for n=1:6
        UB(n) = min(last(n),UB0(n));
    end
end
```

[目标函数]

```
function f4 = ques4sol(v,last,theta)
last(2) = last(2)+v(1);
last(5) = last(5)+v(6);
last = last - v;
x = qiuzhixin(last/850, theta);
f4 = x(1,1)^2 + x(1,2)^2 + x(1,3)^2;
end
```